Engineering framework for scalable recombinase logic operating in living cells

Sarah Guiziou

THESE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER

En Biochimie et biologie pour la santé

École doctorale Sciences Chimiques et Biologiques pour la Santé CBS2 N°168

Unité de recherche Centre de Biochimie Structurale de Montpellier
CNRS UMR5048 – UM – INSERM U1054

Engineering framework for scalable recombinase logic operating in living cells.

Présentée par Sarah GUIZIOU
Le 14 septembre 2018

Sous la direction de Jérôme BONNET

Devant le jury composé de

Dr. Yolanda SCHAERLI, Assistant professor, University of Lausanne	Rapporteur
Dr. Javier MACIA, Professor, Universitat Pompeu Fabra	Rapporteur
Dr. Pascal HERSEN, Directeur de recherche, Université Paris Diderot	Examinateur
Dr. Patrick LEMAIRE, Directeur de recherche, Centre de Recherche de Biochimie Macromoléculaire	Président du jury

UNIVERSITÉ DE MONTPELLIER

Acknowledgments

I would like to thank Prof. Yolanda Schaerli and Prof. Javier Macia for having accepted to evaluate my thesis, I thank you in advance for your feedback on this work. I would like to thank also Dr. Pascal Hersen, Dr. Patrick Lemaire, and Prof. Ricard Solé to act as examiners during my defense. I am looking forward to your striking remarks.

First of all, I would like to warmly thank my PhD supervisor Jerome Bonnet. Jerome gave me a perfect environment to complete my PhD, in terms of science, working environment and people. I am deeply grateful to have him as a supervisor, for his support, his patience, his excitement for science and his sympathy. Jerome guided me during these four years on how to become a good scientist and a good person. Jerome was the best supervisor I could have ever expected, as he is an amazing scientist and more importantly a great person.

I am very grateful to my team colleagues, Hung-Ju Chang, Pauline Mayonove, Peter Voyvodic, Ana Zuniga, Martin Cohen-Gonsaud and Angélique DeVish, which have filled the lab with great energy and good science. It has been great for me to work in this warm and intense scientific atmosphere. I am really thankful to Luca Ciandrini for his great scientific and human support and his calmness at all events. I would like to thank Guillaume Cambray for its helpful scientific critiques and advice. Thank you all for all the friendly times in the lab or around a beer, you all have been of great support: gracias, merci, thanks, grazie, Xiexie. I would like to thank my collaborators at INRA Jouy en Josas, Matthieu Jules and Vincent Sauveplane on the B. subtilis project. It was a pleasure to work with them on this project, and I am thankful for their patience, trust and support. Many thanks also to Nathalie Declerck and Caroline Clerté for having shared their knowledge on the microscope and on B. subtilis with me.

I would like to warmly thank my collaborators at the LIRMM, Michel Leclere, Guillaume Kihli and Federico Ulliana. It was for me a really exciting collaboration, and I am thankful for their excitement for the project, the intense scientific discussion and their patience on our multiple changes of opinion. I am really grateful to Konstantin Todorov who initiated this collaboration, and mainly for his friendship and for having added music to my thesis. I would like to thank Jean-Luc Pons for his patience and help during my linux and programming learning, and thanks also to Laurent Bonnet and Violaine Moreau for the great and warm website that is CALIN.

I would like to thank the interns who worked with me: Léa Meneu, Chloé Tailhades, Sophie Affalo, Stanley Mitchell, Morgane Terezol and Thomas Meiller-Legrand. They all have contributed to this work, they have been great people to work with and they have taught me patience.

A special thanks to the iGEM team 2013. I started to work on recombinase-based logic gates during this summer. Thanks Elodie, Manon, Antoine, Antoine, Quentin, Stick, Yoann,

Fanny, Isabelle, Hang for this intense project. I would like to warmly thank Gilles Truan who supervised us during the summer, had faith in me during this project and has supported me ever since. I would like to thank Claude Marange who took care of all of us during all our degrees.

I am really grateful to have spent these four and a half years at the CBS. It was always a pleasure to come to the lab, thanks to Pierre-Emmanuel Millet who makes this place work as well as it does. Thanks to everyone at the CBS, which has been my second home and family during these years. I am really grateful to have been surrounded by all these skilled, friendly and cheerful colleagues and scientists. I want to especially thank Martin, Angélique, Pauline, Hung-Ju, Peter, Ana, Léa, Aurélie, Michel for being enthusiastic lab neighbors, and for having borne my grumbling. A special thanks to Lucile who guided me through latex and to Ashley who was always open to help me code, write or to have beers. A huge thanks to my office-mates Annika, Pauline, and Anna for the nice and welcoming working atmosphere, it has been wonderful for me to be surrounded by you.

I am deeply thankful to have found awesome people at the CBS with whom I have spent many hours outside of the lab deciphering science and the world. Thanks to Ashley, Mélanie, Elise, Luca, Annika, Lucile, Peter, Pauline, ... all of the beer group.

A warm thanks to all the previous members of the CBS, and especially to my first companions of my thesis, Tom, Solène and Alice.

I would like to give a special thanks to Alexis Courbet, for his friendship, his strong support, and the great philosophical and scientific discussions. Thanks for showing me the path through the PhD.

I would like to warmly thank my two PhD student buddies: Pierre and Carola, with whom I shared all my thesis grumbling and who have always been there in the hard times and have pushed me through to the end. Good luck to Pierre for his last month(s) and to Carola for her final years. I have been lucky during all the years of my thesis and all the years before to have been surrounded by faithful friends. I am really grateful to have you guys, thanks Klervi, Melissa, Helo, Elodie, Caro, Zoé, Xavier, Manon, Antoine, Arthur, Helene and to the little Gauthier.

To finish, I would like to warmly thank my parents, Maurice and Dominique, and my brothers, Léo, Maël, Erwan and Thomas for their unconditional support. A special thanks for welcoming me during my writing retreat, and thanks to Thomas for the gaming breaks. Thanks for your support from the beginning and for having always believed in me.

I want to give a special thanks to sci-hub which has been a precious and loyal help during these four years. Thanks also to the birds and the four hens who have kept me company during my writing. Many thanks to the kilograms of bacteria that I have killed during my thesis; thanks for having given your body to science.

Contents

Chapter 1

Introduction

Contents

1.1 Synthetic Biology

1.1.1 A brief history of synthetic biology

« *Toutes les sciences naturelles suivent une évolution analogue, elles débutent par l'observation et la classification des objets et des phénomènes, puis elles décomposent ceux-ci pour déterminer le mécanisme physique de leur production, elles deviennent alors analytiques ; lorsque le mécanisme d'un phénomène est connu, il devient possible, en dirigeant les forces physiques, de reproduire ce phénomène ; la science est devenue synthétique. [...] La biologie doit évoluer comme les autres sciences naturelles et être successivement descriptive, analytique et synthétique.* »

« *All natural sciences follow an analog evolution, they start from the observation and classification of objects and phenomena, they decompose these to determine the physical mechanism of their production, then they become analytic ; when the mechanism of a phenomenon is known, it becomes possible, by directing the physical forces, to reproduce this phenomenon ; science is becoming synthetic [...] Biology has to evolve as other natural sciences and to be successively descriptive, analytic, and synthetic.* »

Stéphane Leduc 1910

« *What I cannot create I do not understand.* »

Richard Feynman 1988

Since the dawn of civilization, humans have studied and used living organisms that surrounded them, but also shaped those life forms via selective breeding to obtain improved sources of food and materials. We also developed workflows for large scale production of refined foods, such as beer or bread with yeast. However, at this time, we did not understand how living organism operate at the molecular level.

The term "synthetic biology" was used for the first time by Stéphane Leduc [Leduc 1910] [Leduc 1912]. Stéphane Leduc was interested in the synthesis of life from inanimate materials. At this time, he envisioned that biology would progress like the other sciences by successively being descriptive, analytical, and finally synthetic. This idea was mainly inspired from Jacques Loeb and his mechanistic concept of life.

In 1961, in their publication summarizing their study of the lac operon, Jacques Monod and François Jacob envisioned the future ability to assemble new regulatory systems from elementary molecular components [Monod 1961]. This publication is now considered as the origin of synthetic biology [Cameron 2014]. It is also contemporary with the discovery of the structure of DNA [Watson 1953], demonstrated a few years before to be the support of genetic information by Avery and colleagues [Avery 1944] . By the late fifties, Francis Crick introduced the first definition of the central dogma of molecular biology [Crick 1958].

Between 1960s and 1980s, various molecular biology tools were developed, leading to the field of genetic engineering. These breakthroughs included: (1) the discovery of restriction enzymes in the 1960s (Nobel Prize 1978) [Arber 1962, Smith 1973]; (2) the application of the restriction enzymes for DNA recombinant technology (Nobel Prize 1980) [Jackson 1972]; (3) the development of oligonucleotides synthesis [Beaucage 1981, McBride 1983]; and (4) the development of the Polymerase Chain Reaction (PCR) (Nobel Prize 1993) [Mullis 1986].

In 1978, to congratulate Daniel Nathans, Werner Arber, and Hamilton Smith for the Nobel Prize on DNA restriction enzymes, Szybalaski and Skalka wrote, "The work on restriction nucleases not only permits us easily to construct recombinant DNA molecules and to analyse individual genes, but also has led us into the new era of" synthetic biology" where not only existing genes are described and analyzed but also new gene arrangements can be constructed and evaluated" [Szybalski 1978]. All these technological developments quickly enabled the production of proteins of interest using microorganisms, such as recombinant somatostatin and insulin [Itakura 1977] [Goeddel 1979]. However, genetic engineering was mostly restricted to cloning and recombinant gene expression. Detailed knowledge of biological systems was still limited in part due to the technological limitations of DNA sequencing at the time.

In the mid-1990s, the development of high-throughput techniques for DNA sequencing, quantifying RNA, protein, lipids, and metabolites, and the increasing capacities of computational tools led to the field of systems biology. Biologists and computer scientists worked in symbiosis to reverse-engineer cellular networks. From this basic research effort emerged a view of cellular networks organized as a hierarchy of discernible and functional modules [Hartwell 1999].

Stéphane Leduc said: "Biology must evolve like other natural sciences and to be successively descriptive, analytic and synthetic." (1910). As such, the development of systems biology with the view of organisms as composed of modular, regulatory networks laid the foundation for synthetic biology. The construction of new biological systems permits (1) the further understanding of biology and (2) the use of engineered biological systems with novel functions for biotechnological applications, such as manufacturing high-value compounds or addressing unmet healthcare needs.

However, the construction of useful synthetic biological system remained "an expensive, unreliable and ad hoc research process" [Endy 2005]. Engineering biology is indeed a great challenge due to the large complexity of biological systems. To facilitate the engineering of biology, scientists from various backgrounds were inspired from other engineering fields. They applied well-known engineering principles, such as standardization, decoupling, and abstraction, to simplify the construction of synthetic biological systems. The use of these engineering concepts, together with the development and sharing of common tools and platforms is what clearly differentiates genetic engineering from synthetic biology.

The first Synthetic Biology conference, hold at MIT (SB1.0) in 2004, was an important catalyst for the nascent field and helped create its community and culture. Moreover, the

iGEM competition, a synthetic biology student competition each year gathering teams from the entire world (4 teams in 2004 and 337 teams in 2017) has supported the quick, worldwide expansion of synthetic biology, the training of young synthetic biology researchers, and helped spread public awareness of the field.

In order to standardize synthetic biological systems, researchers decomposed them into devices and biological parts. Parts, devices, and systems are supposed to be stored with their precisely and documented characterization in an open-access database such as the Registry of Standard Biological Parts (`http://parts.igem.org/Main_Page`). Shared information should support further construction of more complex systems using already optimized and well-characterized parts. This open-access culture is a strong component of synthetic biology inspired from computer science.

In the past 15 years, many parts and circuits of increasing size have been engineered, leading to a large collection of biological parts and a large set of applications in biotechnology and health (Figure 1.1). However, the complexity of synthetic biological systems did not increase much as expected in the early 2000s. Parts composition is still unreliable and circuit design is still mainly a trial-and-error process. Moreover, the specifications of standard biological parts are still not common to all and little effort is made on the standardization and distribution of well-functioning parts. Due to intellectual property and commercialization concerns, a part of the community does not intend to distribute its work. Consequently, we are now at a critical point where the community has to choose between the *ad-hoc* development of proof-of-concept and prototype circuits and pushing the standardization and characterization of parts and part composition. The challenge is to provide an open ecosystem that supports academic research, companies development, and public access to the tools of synthetic biology. The following years will be crucial.

In my opinion, synthetic biology could be separated in three paths with different, yet complementary, often overlapping goals: (1) the development of foundational technology supporting the engineering of biology: new design and engineering principles, standards, and workflows, including the development of standard parts and devices; (2) the application of these tools to answer fundamental questions in basic research; and (3) the application of these devices to solve pressing challenges in biotechnology, health, and the environment. In my thesis, I developed the first aspect.

In this introduction to synthetic biology, I will present: (1) the engineering principles that founded the synthetic biology community; (2) the techniques enabling high-throughput construction and characterization of biological circuits; (3) the fundamental biological parts and devices which were engineered these past years; and (4) the various applications of synthetic biological circuits.

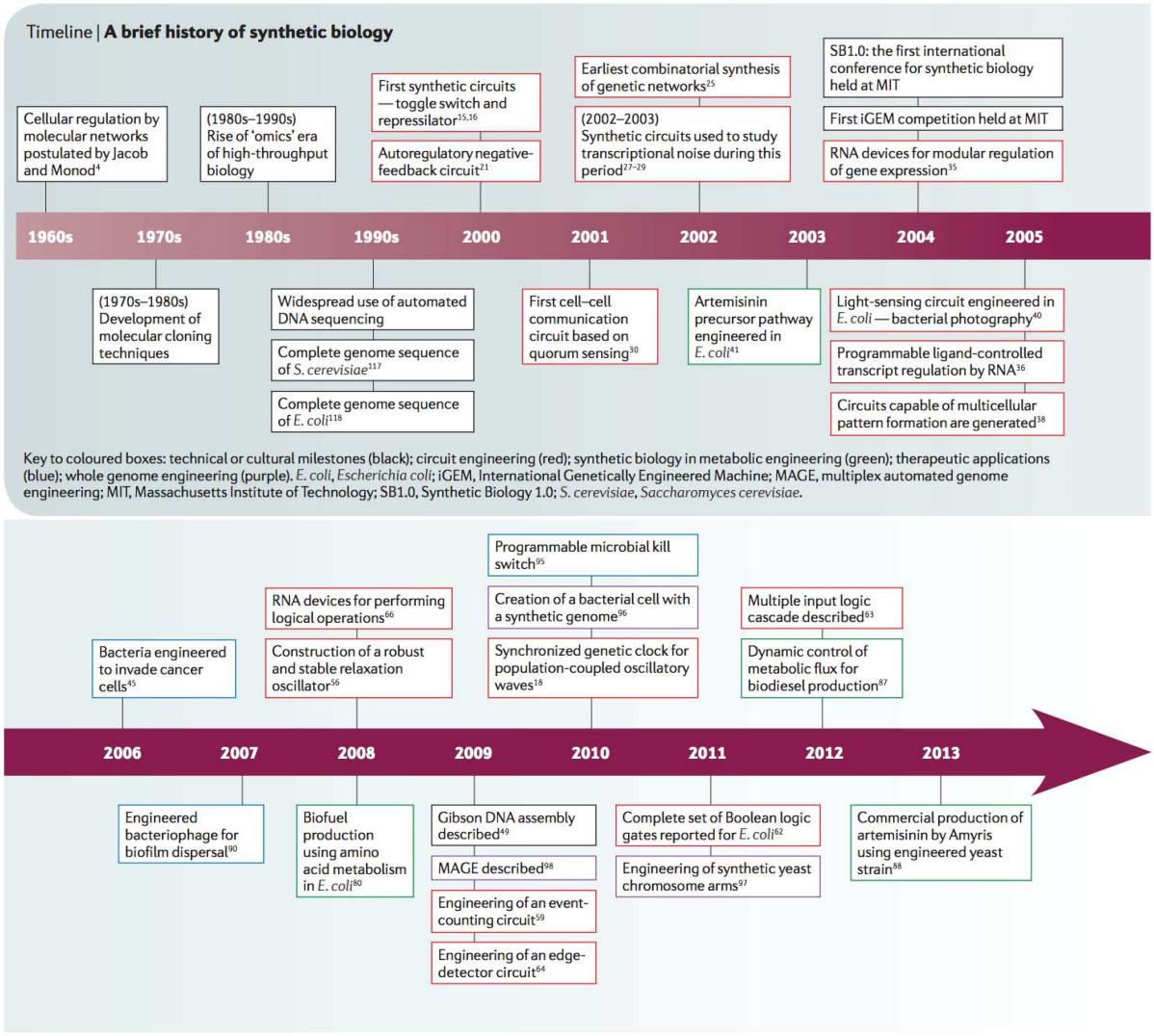


Figure 1.1: A brief history of synthetic biology. Timeline from [Cameron 2014]

1.1.2 Principles of synthetic biology

Synthetic biology is based on engineering principles applied to the construction of artificial biological circuits, networks, and systems. Engineering is defined as a branch of science and technology concerned with the design, building, and use of engines, machines, and structures. It has been previously applied to physics (e.g. in aviation) and chemistry. The three foundations for engineering biology defined in 2005 by Drew Endy are, similarly to other disciplines, standardization, decoupling, and abstraction [Endy 2005]. Standardization corresponds to the definition of standards for biological parts, functions, experimental measurements, system operation, and data-exchange protocols. Decoupling is the decomposition of a complicated problem into many simpler problems that can be worked out independently, such as the decoupling of design and fabrication, and the decomposition of biological systems into simple devices and parts. Abstraction involves the modelization of biological circuits and systems and the organization of biological function information in different levels of complexity that can be manipulated without

detailed knowledge of the lower layers.

The main concept that combines these 3 principles is the parts, devices, and systems approach (Figure 1.2), which can be defined as follows [Baldwin 2015]: (i) Parts encode biological functions (e.g. promoters, terminators, ribosome binding sites, genes). (ii) Devices are made from a composition of parts and encode human defined functions (e.g. logic gates, AHL detectors) (iii) Systems perform complex tasks decomposable in functions, such as counting or intracellular control functions.

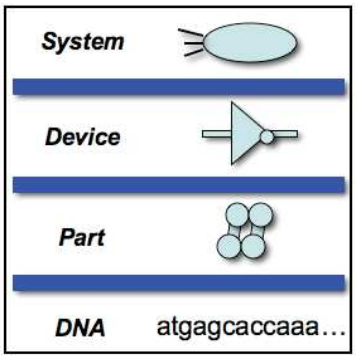


Figure 1.2: Part-device-system principle in synthetic biology.

Great efforts have been put in the definition, standardization, and characterization of biological parts. First, a common language was defined [Arkin 1999], followed by a standard for part assembly [Knight 2003], a datasheet for part description [Canton 2008], and an *in vivo* reference for measuring part activity [Kelly 2009]. These standardizations were broadly deployed to the world community via the iGEM competition, which pushed the young participants to use these standards. Through iGEM, the community has been shaped from the beginning to foster well-characterization and open-access distribution of standard parts [Smolke 2009]. All iGEM teams, and some partner labs deposed their biological parts and devices in the iGEM registry, quickly resulting into a large collection of parts. However, only a small number of these parts are well-characterized and reliable. Consequently, a reduced set of parts are repetitively used. Presently, the iGEM competition continues to emphasize for parts and measurement standardization, for example via the "Measurement" track of the competition.

Most engineering processes are based on the design-build-test cycle (Figure 1.3). Modeling and prediction of system behavior permit the reduction of cycles required to obtain the final circuit. However, for biological circuits, due to the lack of models and the low composability of parts, the design-build-test cycle is at the core of the engineering process and frequently several rounds have to be performed. To speed-up the process, circuit variants are constructed and characterized simultaneously.

We can define the design-build-test cycle following this series of steps:

(i) definition of circuit specifications.

(ii) decomposition of the circuits into simple devices and parts.

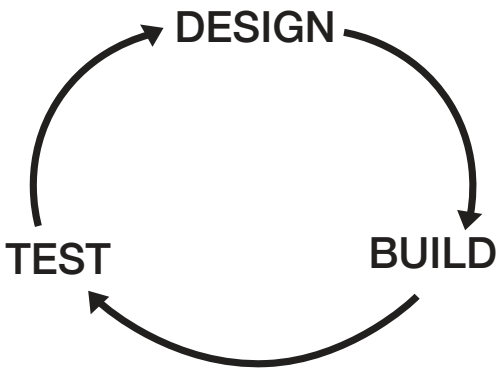


Figure 1.3: Design, build and test cycle.

(iii) design, construction, and characterization of the non-existing parts according to existing databases (design-build-test cycle).

(iv) composition of the parts/devices to build up the circuits.

(v) precise and well-documented measurement of circuit behavior in standard conditions.

(vi) comparison of circuit behavior with circuit specification; leading either to a new cycle with optimization of circuit according to comparison analysis or distribution of the circuit.

The steps from (i) to (ii) corresponds to the DESIGN step. The step (iii) corresponds to a separated design, build and test subcircuit. The BUILD corresponds to partially (iii) but mainly (iv), and TEST to (v).

An important step is the decomposition of the circuit into parts and devices. Each part and device should be characterizable and reusable independently of the full circuits. It permits to simplified characterization processes and to shorten subsequent circuit design.

One important aspect for simplification of biological engineering is the decoupling between design and fabrication. Most genetic engineers were/are limited by the synthesis and assembly of DNA parts. With the rapid development of synthesis, sequencing, and assembly technology, their robotizations, and corresponding price drop, more and more circuit construction will be not be performed by the designer. We will have in the following years a complete decoupling between design and fabrication of genetic circuits.

1.1.3 Technologies underpinning to the development and extension of synthetic biology

Improvement in DNA sequencing, DNA synthesis, computational tools and automation support the development of synthetic biology.

DNA sequencing

The human genome project was completed in 2001 after 13 years and 2.1 billion USD [Shendure 2017]. 17 years later, the genomes of most known organisms have been sequenced and full genomes are sequenced daily for research and health applications. The development of new sequencing methods, along with the advances in computation, triggered a huge drop in price and time required for genome sequencing. In 2008, the development of next-generation sequencing (NGS) brought the price of full-genome sequencing, from around 10M USD to 10K USD in 3 years. Now, full genomes are sequenced for $1,000 in 24 hours, with some technologies even allowing sequencing in one hour.

In synthetic biology, Sanger sequencing is used routinely for verification of DNA cloning. Commonly, samples are sent to service providers where single sequence reactions cost from 2.5 to 5 euros and require 12 hours to 3 days, depending on the type of service requested; however, here to, next-generation sequencing is starting to replace Sanger sequencing.

DNA synthesis

Gene synthesis is based on the synthesis of an oligonucleotide pool that are then combined into double-stranded DNA. The drop in the cost of oligonucleotide synthesis has been a crucial step in the development of affordable DNA synthesis. As of today, an oligonucleotide of around 20 bp costs around 2.50 euros.

The cost of double-stranded DNA recently dropped with the development of silicon-powered DNA synthesis, which miniaturized the synthesis process. For example, the synthesis of double-stranded DNA costs $0.07 per bp within 7-10 working days, and $0.09 per bp if coupled with cloning within vector (20 working days). Faster synthesis is also possible, such as synthesis and shipping of short double-stranded DNA of less than 750 bp in 2 to 4 business days.

The low price and short time of synthesis has changed our way of doing synthetic biology, as the limited step is mainly the design and characterization of new synthetic circuits and not their construction. However, for large circuits or specific vectors, cloning is still required as the length of synthesizable fragments is limited and some low-copy plasmids not supported by cloning platforms.

DNA assembly

The assembly of DNA parts has always been a crucial step in synthetic biology workflows, as circuits are assembled from various parts, either pre-existing or synthesized. To facilitate DNA assembly, a standard assembly process based on restriction enzymes was developed to allow low-cost and systematic assembly of synthetic devices. The Biobrick Standard Assembly developed in 2003 by Tom Knight is based on 4 restriction enzymes: EcoRI, SpeI, NheI, PstI [Knight 2003]. All parts are surrounded by a prefix composed of EcoRI and SpeI and a suffix composed of NheI and PstI. The assembly of two Biobrick parts leads to a Biobrick part still composed of a prefix and a suffix, as the two previous parts are assembled via a 6 bp scar formed by SpeI

and NheI. Assemblies based on restriction-ligation, such as Biobrick assembly, are not very efficient and require standardization of parts by removing the restriction sites naturally existing in the parts. Other assembly methods which are modular, single-step, and more efficient were developed, such as Gibson Assembly [Gibson 2009] and Golden Gate Assembly [Engler 2008]. The Gibson Assembly method that I mainly used during the last 4 years uses a 40 bp sequence overlap between assembled DNA fragments using a single isothermal step, where the sequence homology between fragments can easily be added by PCR. This method permits assembly of up to 5 fragments in a single experiment.

Due to the decrease of synthesis prices, DNA assembly, strain construction, and characterization are increasingly performed at medium- or high-throughput rates (several tens to hundreds per day). However, repetitive manual pipetting steps are required which take time and lead to large error probabilities. Pipetting robots are now available at low cost and with reduced programming requirements. One example is the Opentrons robot composed of one to two pipetting arms and a simple and flexible programming interface via Python, available for $4,000. More complex robots have been also available for decades but required experts for their programming, which make them useful for companies to automatized processes but less useful for academic labs. Moreover, several companies have now created robot facilities to perform experiments such as high-throughput cultures, flow-cytometer and plate reader analysis. Samples, strains, and experiment specifications are sent to the facilities and all experiments are performed by the outsourced company. While not yetwidely used, soon all experiments will be performed by service providers using automated pipelines and researchers will be able to focus only on experiment design and result analysis. Researchers will thus spend more time thinking about their design than performing repetitive steps; although experiments requiring high-level of precision or not amenable to automation will still be performed by the researcher.

This workflow corresponds to a totally decoupled design-build-test cycle with highly specialized engineers at each step.

1.1.4 Engineering part libraries and complex devices

Synthetic biology is based on the engineering of genetic circuits composed of standard biological parts. Therefore, for the precise engineering of complex circuits, large libraries of biological parts enabling fine tuning of gene expression are needed. In response to these needs, several libraries of components were engineered to regulate gene expression at several levels (mainly transcription and translation) for many organisms of interest including *Escherichia coli* [Mutalik 2013a], *Saccharomyces cerevisiae* [Lee 2015], and mammalian cells [Ede 2016].

Many part libraries were developed for the Gram-negative bacteria model *E. coli*, as it has historically been the most widely used organism in synthetic biology. Such libraries include collections of promoters, terminators, ribosome binding sites, and repressors for the regulation

of gene expression named GOI (Gene Of Interest). The objective of these libraries is to have sequence variants and fine-tuning of gene expression at various levels. Sequence variants are essential for the engineering of large genetic circuits to avoid recombination between homologous parts [Nielsen 2016]. The BIOFAB (International Open Facility Advancing Biotechnology) foundation aimed to the design and build of biological parts. The BIOFAB was indeed at the origin of the construction and characterization of large libraries of promoters [Kosuri 2013] [Mutalik 2013a], RBSs [Gardner 2000] [Egbert 2012] [Kosuri 2013] [Mutalik 2013a] and transcription terminators [Chen 2013] [Cambray 2013].

At the beginning of my Ph. D., I performed similar work for the Gram-positive bacteria model *B. subtilis* [Guiziou 2016]. The number and diversity of biological parts for *B. subtilis* was limited at this time, despite its long history as a model organism and biotechnology workhorse. Therefore, I engineered libraries of promoters, ribosome binding sites, and degradation tags. For clarity and flow, I will not detail this work later on my manuscript; however summary of this work is available in the following text box and a reprint of the publication can be found in Annex C.

A part toolbox to tune genetic expression in *Bacillus subtilis*.

Sarah Guiziou, Vincent Sauveplane, Hung-Ju Chang, Caroline Clerte, Nathalie Declerck, Matthieu Jules and Jerome Bonnet. NAR 2016. [Guiziou 2016]

Libraries of well-characterised components regulating gene expression levels are essential to many synthetic biology applications. While widely available for the Gram-negative model bacterium *Escherichia coli*, such libraries were lacking for the Gram-positive model *Bacillus subtilis*, a key organism for basic research and biotechnological applications. Here, we engineered a genetic toolbox comprising libraries of promoters, Ribosome Binding Sites (RBS), and protein degradation tags to precisely tune gene expression in *B. subtilis*.

We first designed a modular Expression Operating Unit (EOU) facilitating part assemblies and modifications, and providing a standard genetic context for gene circuit implementations. In the Expression Operating Unit, 40 bp spacers are placed at strategic positions to simplify Gibson Assembly.

We then selected native constitutive promoters of *B. subtilis* and efficient RBS sequences from which we engineered three promoters and three RBS sequence libraries exhibiting ~14,000-fold dynamic range in gene expression level. Libraries are generated by randomization of 3 nucleotides in the -10 box or between the -10 and -35 box for promoters and 6 nucleotides in the Shine Dalgarno for RBS. After cloning and integration within the *B. subtilis* genome, libraries were sorted into different bins to obtain different ranges of expression levels. After characterization of a reduced number of variants per bin, we obtained a library spanning the full range of gene expression.

We also designed a collection of SsrA proteolysis tags of variable strengths by randomization of the three C-terminal aminoacids of the *B. subtilis* Ssra tag. Finally, by using fluorescence fluctuation methods coupled with two-photon microscopy, we quantified the absolute concentration of GFP in a subset of strains from the library. Using this subset of absolute quantification, we estimated the absolute concentrations of GFP for our full library.

Our complete promoter and RBS sequence library comprising over 135 constructs enables tuning the GFP concentration in over five orders of magnitude, from 0.05 to 700 μM. This toolbox of regulatory components will support many research and engineering applications in *B. subtilis*. All strains and plasmids are available at BGSC (Bacillus Genetic Stock Center: `http://www.bgsc.org`) and have already been requested many times.

In addition to parts for constitutive expression of genes, inducible parts are needed to permit construction of dynamic circuits and for detection of output molecules. The most used part for inducible expression are inducible promoters based on repressor and activator proteins, such as promoters inducible by isopropyl β-D-1-thiogalactopyranoside (IPTG), anhydrotetracycline (aTc) and arabinose. More details on parts for inducible expression of gene are available in the Section 2 describing the implementation of logic in living organisms.

The construction of these large part libraries have shown that gene expression is highly dependent on genetic context. In other words, the behavior of a particular combination of promoter, 5'UTR and gene cannot be inferred from the separated behavior of each part. Mutalik and colleagues quantified part reliability, and found that transcription and translation efficiencies are not only dependent on promoter and ribosome binding site efficiencies but also on interaction between parts such as promoter and GOI [Mutalik 2013a]. Consequently, methods and tools were developed to insulate parts from genetic context, such as the use of BCD [Mutalik 2013b], ribozymes [Lou 2012], terminator upstream of gene expression cassettes, and insulated promoters [Davis 2011]. Insulation of parts permits the reduction of the trial-and-error process for engineering large gene regulatory networks [Nielsen 2016] [Zong 2017].

Based on these parts regulating gene expression, more complex circuits were built. 2 gene regulatory circuits, the repressilator of Elowitz and Leibler [Elowitz 2000], and the toggle switch of Gardner and Collins [Gardner 2000] are considered to be the starting point of modern synthetic biology. Of note, a precursor circuit, similar to the Gardner et al. toggle switch, was constructed in 1985 [Toman 1985] for the detection of alterations in *Escherichia coli* induced by DNA-damaging agents. Both circuits from 2000 are based on repressors cross-repressing their transcriptions. For the two papers, a simple mathematical model of circuit behavior was defined and permitted the identification of the parameters and conditions to tune circuit behavior. During the same period, cell-cell communication circuits were developed based on quorum sensing using AHL, LuxI, and LuxR system [Weiss 2001]. These circuits have largely been reused, optimized, and improved in the 15 following years. A diversity of gene regulatory circuits exists nowadays. Some circuits are now part of synthetic biology textbooks such as pattern formation [Basu 2005], edge-detector [Tabor 2009], predator-prey [Balagaddé 2008] and integrase-based Boolean logic gate circuits [Bonnet 2013] [Siuti 2013].

All of these circuits were engineered for *E. coli*; indeed, most part libraries and devices are designed and characterized for *E. coli*. However, tools for other organisms are now being developed.

As we still have a very limited understanding of how biological systems work, rational design of biological circuits and molecules, especially proteins, remains a tedious and inefficient task. Nature evolved during billions of years to lead to today's complex organisms. Evolution has been mimicked in lab through randomization and selection, accelerating evolution time scale to a few days or weeks. This process, called directed evolution, therefore permits the testing of potential

nature diversity, and using a well-designed selection process to obtain the biological circuit of interest for a desired behavior [Arnold 1998]. Directed evolution is performed at different levels, such as the level of proteins, networks, or full organisms. The engineering of proteins through directed evolution has been performed for decades to improve the catalytic activity of proteins and to design *de novo* catalytic activities [Arnold 1999, Farinas 2001]. Additionally, directed evolution was used to obtain new regulatory elements. By combining evolution of proteins and regulatory elements, networks and metabolic pathways of increasing efficiency have been engineered [Umeno 2004]. In a general manner, error-prone PCR or randomized oligonucleotides are used to generate diversity. For evolution of full organisms, the multiplex automated genome engineering (MAGE) method was developed [Wang 2009], permitting to development of ideal host organisms [Cobb 2013].

1.1.5 Applications of synthetic biology

Synthetic systems are engineered for three different purposes: (1) the development of tools to engineer biology, (2) the study of biology and (3) its application to actual challenges. I previously reviewed tools developed for the engineering of biology. A reduced number of systems are directly applied to fundamental biology, and most of them are used in the construction of synthetic organisms, mainly synthetic and minimal genomes. Regarding applications, synthetic biology is broadly used for (a) the production of molecules of interest (metabolic engineering), (b) healthcare via the development of cell-based therapeutics and diagnostic systems, (c) bioremediation and biomaterial production, and (d) art & design.

1.1.5.1 Synthetic biology to build synthetic organisms

The development of synthesis and assembly strategies allowed the synthesis of entire eukaryotic genomes, like *Mycoplasma capricolum* or most of the *S. cerevisiae* chromosomes. In fact, most assembly strategies, such as Gibson Assembly, were developed for the purpose of assembling large genome fragments.

The first synthetic genome was that of the bacteriophage ϕX174 (5,386 bp) in 2003 [Smith 2003], followed by the "small" bacterial genome of *Mycoplasma genitalium* (582,970 bp) [Gibson 2008]. The synthesis of an entire functional genome became more realistic with the synthesis of the 1 Mbp *Mycoplasma mycoides* genome JCVI-syn1.0 and its transplantation in *Mycoplasma capricolum* [Gibson 2010]. In addition to synthesizing natural genomes, the feasibility of whole-genome scale recoding was shown by swapping all rare codons in 42 genes in *E. coli* [Lajoie 2013a]. Also, a full genomically recoded organism (GRO) was generated by replacing all known TAG stop codons of the *E. coli* MG1655 [Lajoie 2013b] and expanded to an *E. coli* genome in which seven codons have been replaced [Ostrov 2016].

The Synthetic Yeast Genome Project (Sc2.0) started in 2006. The project aims to synthe-

size of the full *S. cerevisiae* genome, composed of 16 chromosomes and 12.5 Mbp. The first chromosome was finished in 2014 [Annaluru 2014]. The design principles of this large scale genome were: (1) to have a genome produce a phenotype similar to the wild type, (2) to rearrange tRNA genes, introns, and transposons to improve the genome stability (all tRNA genes are positioned on another chromosome) and (3) to have genetic flexibility to facilitate future research. The third point was addressed by changing all TAG stop codons to TAA, by adding short PCR-Tags to permit distinction between synthetic and natural genome, and by adding loxPsym sites that permit synthetic chromosomal rearrangement and modification via loxP-mediated evolution (SCRaMbLE) [Shen 2016]. In 2018, 7 chromosomes have been successfully synthesized [Richardson 2017] and 3 synthetic chromosomes have been placed in a single yeast cell capable of conserving its wild type fitness. The SCRaMbLE technique has been used to study chromosomal structure [Shen 2016] [Mercy 2017]. This synthetic approach could be useful to study chromosomal behavior and to generate a minimal yeast genome for industrial production.

After synthesizing the smallest known cultivable bacterial genome, Craig Venter's group pursued their efforts to obtain a minimal genome [Hutchison 2016]. A minimal genome was designed using Tn5 transposon mutagenesis data [Glass 2006]. After four rounds of design-build-test cycles, they divided the size of the *Mycoplasma genitalium* genome by two: from 1079 kb for the LCVI-1.0 genome template to 531 kb for the JCVI-syn3.0 genome encoding 473 genes. Surprisingly, 30% of essential or quasi-essential genes have no known function. Therefore, this minimal *Mycoplasma genitalium* genome is a great tool to understand the essential mechanisms of life.

Despite our large scale synthesis capacity, much work remains before genomes can be designed from scratch; the more we understand biological networks, the more we realize how little we do know. These synthetic and simply-modifiable genomes will extend our understanding of genome structures and functions.

Researchers are also interested in building synthetic cells from scratch [Gopfrich 2018]. Cooperative projects are starting with the objective of building an autonomous, self-replicating cell (http://www.basyc.nl/, Basyc (Building a Synthetic Cell); Max-Plank). Projects are divided in modular building blocks, such as fueling, DNA processing, and cell division. In addition to the potential applications from this large cooperative effort, it will allow researchers to better understand the origins of and requirements for the emergence of self-replicating organisms.

1.1.5.2 Synthetic biology methods for bio-production and manufacturing

With increasing frequency, valuable molecules are being synthesized using living organisms. Indeed, many molecules are naturally produced by living organisms, and these specific metabolic pathways can be implemented in industrial organisms, such as *S. cerevisiae*, to obtain high-yield

production via fermentation. These living organisms have been used for industrial production centuries before synthetic biology, such as for the production of beer. For a few decades, metabolic engineering has consisted mainly in the optimization of natural metabolic pathways or the addition of a single enzyme to obtain a new biochemical compounds. Now, full metabolic pathways are implemented in living organisms. The most famous example is the production of the artemisinic acid in *S. cerevisiae* via the implementation of a synthetic metabolic pathways composed of 4 new genes and 7 up- or down-regulated ones [Ro 2006] [Paddon 2013]. This metabolic pathway, coupled with a chemical transformation, produces industrially relevant concentrations of artemisinin: a drug essential for malaria prevention, and the process has been commercialized by Sanofi.

Additionally, complete biosynthesis pathways of opioids were engineered in yeast. Opioids are high values compounds and the primary drugs for pain management and palliative care. Galanie and colleagues focused on the production of thebaine and hydrocodone, which required expression of respectively 21 and 23 enzymes originating from plants, mammalians, bacteria, and yeast itself [Galanie 2015]). This synthetic metabolic pathway could be tailored to produce other natural opioids, as well as intermediate compounds rarely accumulated in plants, but which potentially have interesting pharmaceutical properties. This first proof-of-concept did not show a high production yield, however, the pathways is currently being optimized to fit the requirements for industrialisation. As opioids are high-value components, the yield required to obtain a profitable process is low and therefore more easily attainable using living organisms and current synthetic biology techniques.

In another field of application, Schwander and colleagues designed and constructed a synthetic pathway for the conversion of CO_2 into other organic molecules [Schwander 2016]. This CETCH cycle (crotonyl-coenzyme A (CoA)/ethylmalonyl-CoA/hydroxybutyryl-CoA cycle) was designed by metabolic retrosynthesis and is composed of 17 enzymes originating from nine different organisms plus three reactions created by rational active-site engineering. This large pathway adds a synthetic alternative to the six CO_2 fixation pathways identified in nature.

To extend the range of molecules synthesizable by living organisms, the production of five different acyl-CoA esters was developed in *S. cerevisiae* by Krink-Koutsoubelis and colleagues. The acylCoAs produced in this study are common building blocks for secondary metabolites and will enable the engineering of the production of a variety of natural products in *S. cerevisiae* [Krink-Koutsoubelis 2018].

A lot of works has been performed on the use of microorganisms to replace dependency on petroleum, such as for fuels and plastics. However, the production of these materials of commodity has to be competitive with the cost of drilling and refining petroleum. Ethanol, butanol, pentanol, propanol and derivative productions have been optimized in microorganisms [Lee 2008], but the yield still remains too low to compete with petroleum.

Large metabolic pathways for medium- or low-value components should be assembled soon

with the expansion of metabolic engineering building blocks and the development of *de novo* enzyme engineering.

1.1.5.3 Synthetic biology for environmental and healthcare applications

Environmental biosensing and bioremediation

With the extension of the human population and industrialisation, we are polluting the soil and water at large scale. It is thus essential to develop tools to monitor our environment. Moreover, the study of how microorganisms composing our environment react to changes of environmental conditions is a great challenge. To tackle this problem, both analytical chemistry and whole-cell biosensors have been developed; here, I will focus on whole-cell biosensors. Fulfilling most of the technical requirements of analytical chemistry such as high specificity, sensitivity, and reproducibility, whole-cell biosensors additionally provide rapid responses, simple preparation methodologies and cheap detection systems [Renella 2016].

Natural biosensors, such as bacteria (e.g. *Vibrio fischeri*) or micro-eukaryotes are used to analyse the global toxicity of environments. Similarly, synthetic biosensors producing luciferase permit detection by luminescence to quantify the cell integrity and full metabolic activity as luminescence is reduced upon cell damage or toxicity.

Additionally, synthetic biosensors responding to specific molecules were engineered by placing GFP or luciferase gene expression under control of natural promoters responding to chemical components. To date, a variety of target analytes such as organic xenobiotics (naphthalene, BTEX [benzene, toluene, ethylbenzene and xylene], alkyl-sulphonates, and polychlorinated biphenyls), heavy metals and metalloids (As, Cd, Zn, Ni, Cu, Cr, Cu), nutrients, and physiologically active molecules can be detected by different kinds of whole-cell biosensors [Renella 2016].

Whole-cell biosensors permit the analysis of not only the concentration of pollutant in the environment but also its bioavailability, bioaccumulation, and biomagnification as the selected analyte needs to cross the cell membrane for detection. For now, the use of these detection systems is mainly limited to research, but it should soon be used in the official methods for soil and environmental analysis.

Microorganisms have also been used for their natural capacities to consume or breakdown polluant [Vidali 2001]. By combining these natural bioremediation capacities with biosensing, the bioremediation efficiency of microorganisms can be increased [de Lorenzo 2008].

Applying synthetic biology to healthcare.

Many efforts have been made to use synthetic biology in the clinics and for the development of diagnostic and therapeutic systems [Courbet 2015a, Ruder 2011]. These efforts include the development of synthetic biology therapies for the treatment of infectious diseases and cancer, as well as portable diagnostic devices, cell therapy, vaccine development, and microbiome

engineering.

As numerous applications of synthetic biology to healthcare exist, I will focus here on a few representative examples, such as (1) cell therapy with engineered CAR T-cell, (2) engineering of bacteria for diagnostics, (3) engineering the microbiota to fight infectious disease and cancer.

For cancer therapy, the most advanced synthetic-biology based therapy is using CAR T-cells, where recently being to be commercialized by Novartis. CAR T-cells are based on chimeric antigen receptors (CARs) composed of antibody-binding domains fused to T-cell signaling domains [Kalos 2013]. The therapeutic process consists of retrieve T-lymphocytes from patients, re-engineering them *ex vivo*, and reinfusing a large quantity of the engineered T-lymphocytes into the patient. This CAR T-cell strategy is applied to chemotherapy-resistant leukemias and in a number of cases has led to complete and long-lasting clinical remission.

Microorganisms have also been used as biosensors for diagnostic, monitoring, and epidemiology [Chang 2017]. Bacteria sensing quorum-sensing molecules were engineered for detection of infection [Kumari 2008], speeding up the identification of infectious agents. Using yeast, antibody display was used to perform electrochemical detection of *Salmonella* or the Hepatitis C virus [Aronoff-Spencer 2016]. Going beyond living cells, cell-free systems have been developed for detecting of nucleic acids from Ebola or Zika virus [Pardee 2014][Pardee 2016]. Diagnosis using microorganisms can be performed *ex vivo* but also *in vivo*. Indeed, the human microbiome outnumbers the human cells by a factor of 10 to 100 and play an essential role for our overall health. Consequently, microbiome engineering is a prime area for diagnosis and therapeutic purposes. One example is the engineering of the gut bacteria *E. coli* to prevent cholera infection by producing the quorum sensing molecules AI-2 and CAI-1 that repress *V. cholera* virulence [Duan 2010]. Alternatively, bacteria could be engineered to deliver therapeutic molecules directly inside the body and specifically to the disease location. Bacteria have been engineered to specifically migrate to and target solid tumors, reproducing toxins upon arrival to destroy the tumor [Xiang 2006, Anderson 2006].

Numerous studies have been done on engineering living organisms for healthcare, but it is still a challenge to engineer a system which is precise and specific enough to pass clinical trials. The most short-term doable work is probably the development of portable diagnostic methods.

1.1.5.4 From designing biology to biology for design

Living organisms produce very strong materials which can be used and functionalized for human purposes. One of the most studied biomaterial is spider silk, which is used for biotechnological applications such as stem-cell tissue engineering [Wang 2006]. Recently, recombinant silk production has been succesfuly implemented in *E. coli* [Bryksin 2014] [Jiang 2018]. The use of biomaterials produced by microorganisms allows on-demand generation of precise and functionalized nanostructure (Figure 1.4).

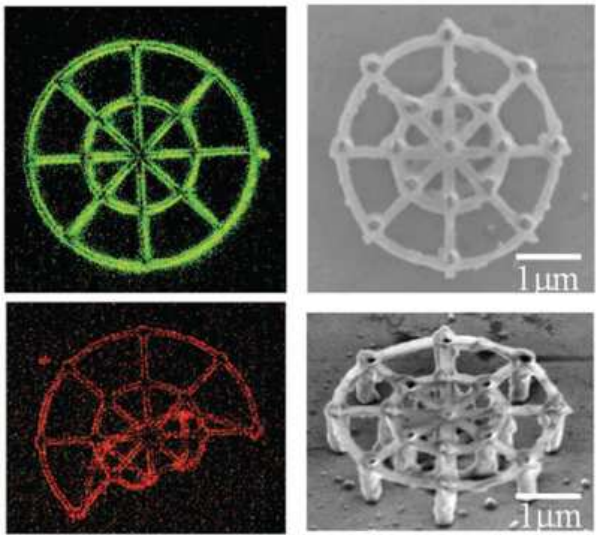


Figure 1.4: **Manufacturing and assembly of 3D bionanoarchitectures via Protein Bricks using IEBL.** Figure from [Jiang 2018]: the fluorescent (left) and SEM (right) images of 3D spider nanowebs via IEBL on fluorescein sodium (green)-doped and RB (red)-doped spider silk. Note that, for the RB-doped spider web, two of the anchoring points were intentionally neglected such that the unsupported part can fold during the water development, showing the 3D nature of the fabricated structure.

Another example of biomaterial production using synthetic biology is a nacre-inspired composite material, which is stronger than cement, obtained through the combination of ureolytic bacteria and bacterially produced γ-polyglutamate (PGA) [Schmieden 2016]. These various provided an example of the potential of biotechnological production of various materials of interest with the applications ranging from biomedecine to architecture and construction.

Bacteria have also been used to produce bioink [Lehner 2017]. Many artists have been using synthetic biology tools for art creation (`http://www.syntheticaesthetics.org/`, [Boland 2013], AnnexH). These paths are of great interest to (1) expand the synthetic biology community and our work and to (2) design the future synthetic biology application with expert designers.

1.2 Logic circuits built using biological components.

All living organisms, from humans to bacteria, sense their environment and their internal state, process this information, and perform specific actions. The detection of changes in pressure, temperature, pH, or concentration of chemicals is essential for living organisms to adapt to environmental changes and to control their internal machinery. A cell can indeed be viewed as a machine for protein production [Jacob 1961], which constantly adapts its behavior according to its environment.

Cells use diverse sensing systems to detect a multitude of signals and respond appropriately. Signal processing is performed through regulatory networks controlling phenotypic responses, like change in metabolism, apoptosis, or cell growth.

In synthetic biology, we aim to implement new or rewire existing networks within living organisms. For this purpose, the development of synthetic detection and computation systems are two important challenges. Here, I will describe fundamental concepts in computation and how they can be implemented within living organisms.

Computation is a general term for any type of information processing, from the human thinking to electronic calculations. It follows a well-defined model which can be expressed in an algorithm.

The interest in implementing computation in living systems focuses on the natural capacities of living organisms to process signals. Living systems are highly complex, more than any system constructed currently by humans. This incredible complexity of nature built over four billion years of evolution gives us the possibility to build highly complex, sensitive, and adaptable computing systems. However, computing with and within living systems does not specially attempt to compete with electronic systems, which are much faster and more adapted for certain kinds of tasks. Bringing human-controlled computational power to biology will serve new purposes for which "smart" engineered biological systems are uniquely suited. As of today, most of the prospects of biocomputing have not yet been even envisioned [Endy 2011].

Nevertheless, because of its success, electronics has provided a robust foundation to implement computing within living organisms. Most electronic devices, such as smartphones, operate using Boolean logic gates. Researchers were inspired from these designs and started implementing circuits made up of biological molecules to perform Boolean logic functions. Inputs for the computing system are environmental stimuli, and outputs mainly consist of gene expression controls leading to a specific cell response or to the expression of a visual reporter.

In this chapter I will present: (1) the foundations of logic and electronic logic designs, (2) *in vitro*, and (3) *in vivo* implementation of logic through biological systems.

1.2.1 Introduction to logic and circuit design strategies

1.2.1.1 A brief history of logic and its application in electronics

Logic was one of the first major philosophical disciplines, the term having been first used by Xenocrates, a disciple of Aristotle in the 4th century B.C.E.. Logic means at the same time reason, language, and reasoning and corresponded at this time to the study of the formal rules that have to follow well-formed arguments.

Aristotle and syllogism [Aristotle] were for two milenia the principal reference in logic. In the XVII century, Godfried Von Leibniz was the first to develop a completely formal logic system. He used logic to formulate ideas, stating that "ideas are compounded from a very small number of simple ideas, and complex ideas proceed from uniform and symmetrical combination of these simple ideas, analogous to arithmetical multiplication" (from Leibniz unpublished work) [Couturat 1901, Couturat 1911]. Leibniz formulated the central concepts of mathematical logic still in use today, such as conjunction (AND), disjunction (OR), negation, identity, and implication.

Furthermore, Leibniz developed the modern binary number system, inspired from the Song Dynasty scholar Shao Yong (1011-1077). The Shao Yong's square, dating from the 11th century in China, represents the numbers from 0 to 63 in 6 lines with either a broken line for the yin (0), or a full line for the yang (1), which is highly similar to modern binary numbers [Arrault 2000] (Figure 1.5).

In the 19th century, George Boole published two groundbreaking books: "Mathematical Analysis of Logic" [Boole 1854, Boole 1847] and "The Law of Thought" [Boole 1854, Boole 1847]. In these works, he proposed a novel logic algebra, now called the Boolean algebra. His fundamental idea was that logical relations could be expressed in algebraic formulae.

He defined in his first proposition of the Law of Thought:

> 1st. Literal symbols, as x, y ... representing things as subjects of our conceptions.
>
> 2nd. Signs of operations, as +, -, x, standing for those operations of the mind by which the conceptions of things are combined or resolved so as to form new conceptions involving the same elements.
>
> 3rd. The sign of identity, =.
>
> And these symbols of Logic are in their use subject to definite laws, partly agreeing with and partly differing from the laws of the corresponding symbols in the science of Algebra.

In 1937, Claude Shannon produced during his master's thesis the conceptual breakthrough that laid the foundation for the modern computing revolution. Shannon transposed Boolean algebra to electronic circuits designs and developed symbolic relay analysis.

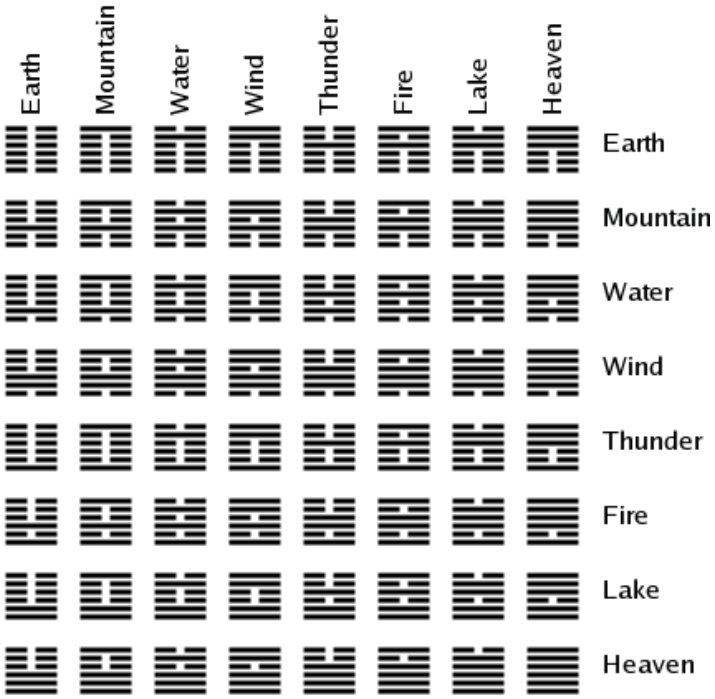


Figure 1.5: The Shao Young square. These symbols date from the 11th century and have an important symbolism in Chinese culture. A broken line corresponds to the yin (0) and a full line to the yang (1). Each symbol composed of 6 lines has a specific meaning in Chinese culture. Leibniz drew inspiration from these symbols to develop modern binary number systems.

In Shannon's symbolic relay analysis, circuits have only two possible states, either 0 for closed circuits or 1 for open circuits. Connection in series of the circuit X and Y corresponds to the sum of X and Y (X+Y) and the connection in parallel to the product (X.Y).

Using this analogy to Boolean algebra, Claude Shannon developed a systematic framework for circuit design. The main philosophy of his design strategy is best summarized in his own words: "For the synthesis problem the desired characteristics are first written as a system of equations, and the equations are then manipulated into the form representing the simplest circuit. The circuit may then be immediately drawn from the equations." [Shannon 1936] The design of modern electronic circuits is based on algebraic logic and switching theory developed by Shannon.

Another breakthrough came with the development of the transistor (John Bardeen) in the 1950s. The transistor enabled the exponential development of electronics. A transistor is an electronic component in which an input stimulus can either be closed or opened as a "valve", allowing or preventing the flow of electrical current. A simple transistor allows implementation of an inverter circuit. The simple logic gates, e.g. NOR and NAND gates, were implemented by placing two transistors either in series or in parallel. The first silicon transistor was commercialized by Texas Instruments in 1954 and 250 billion billion transistors were made in the 2014 year. The number of transistors in an integrated circuit has followed Moore's law, as it doubles

approximately every two years, due to manufacturing innovation and miniaturization. Modern electronic devices are still mainly composed of transistors. The researchers that developed the transistor were awarded the Nobel Prize in physics in 1956.

For this brief history of logic, and during the remainder of my thesis, I focus my work on propositional logic. However, other classes of logic exist, such as predicate, modal, and fuzzy logic. Propositional and predicate logics were developed simultaneously. Propositional logic corresponds to the study of propositions formed by combination of statements with the use of logic operators (OR, AND, NOT). Propositional logic is also called Boolean logic, as it uses Boolean algebra. Predicate logic is commonly used to define mathematical theorems. It extends propositional logic operators with quantifiers such as $\forall$ ("for all"), $\exists$ ("there exists"), and relations such as implication and biconditional. Modal logic was developed on in the 1960s and extended propositional and predicate logic with operators expressing modality, such as necessarily and possibly. Apart from propositional, predicate, and modal logic, fuzzy logic has been studied since the 1920s. In fuzzy logic, variables are not either True or False, but an infinite number of degrees of truth from 0 to 1 are possible.

1.2.1.2 Introduction to combinational logic, Boolean algebra, circuit minimization and design strategy in electronics

Among propositional logic classes, I focus here on combinational logic. This is the class of logic implemented by Boolean circuits where the output is a function of the presence of inputs. Combinational logic circuits can be defined by a truth table, which lists all the possible combinations of input values (True or False, $2^{number\ of\ inputs}$) and associates for each the desired output value (True or False). A truth table can be converted into a truth function, which takes a specific number of truth values as inputs and produces a truth value as output.

In a truth function, used in classical propositional logic, logical operator connecting statements are either disjunction (OR), conjunction (AND), or negation (NOT). The following symbols are used: $\vee$ (disjunction), $\wedge$ (conjunction), $\neg$ (negation).

Boolean algebra was defined to permit simplification of logic equations using common algebra. Consequently, direct conversion from truth function to Boolean function can be performed following Table 1.1.

In electronics, Boolean algebra is more commonly used. However, depending on the field, various terms and symbols are used to write down truth function. In the bio-computation field, different symbols are used depending on the background of the authors and this can be confusing. I will then introduce all possible terms and symbols that one might encounter.

Variables of a truth function are called variable, input, or signal. A literal defines a truth variable or its negation. For the operators of a Boolean and truth function, many different notations and terms are used:

Truth function	Boolean function
True	1
False	0
$\vee$	$+$
$\wedge$	$\cdot$
$\neg$	~

Table 1.1: **Conversion from truth function to Boolean function.**

- For the negation of terms: $\neg$, ~ , ! , /, and NOT symbols are used (Figure 1.6A)

- For the disjunction: $\vee$, $+$, //, and OR symbols are used. Disjunction is also called "sum of terms", as it is series of literal related by OR (Figure1.6B).

- For the conjunction: $\wedge$, $\cdot$, . , &, and AND symbols are used. Conjunction is also called "product of terms", as it is a series of literals related by AND (Figure 1.6C).

Venn diagrams are used to visualise logic operators and were first defined in set theory.

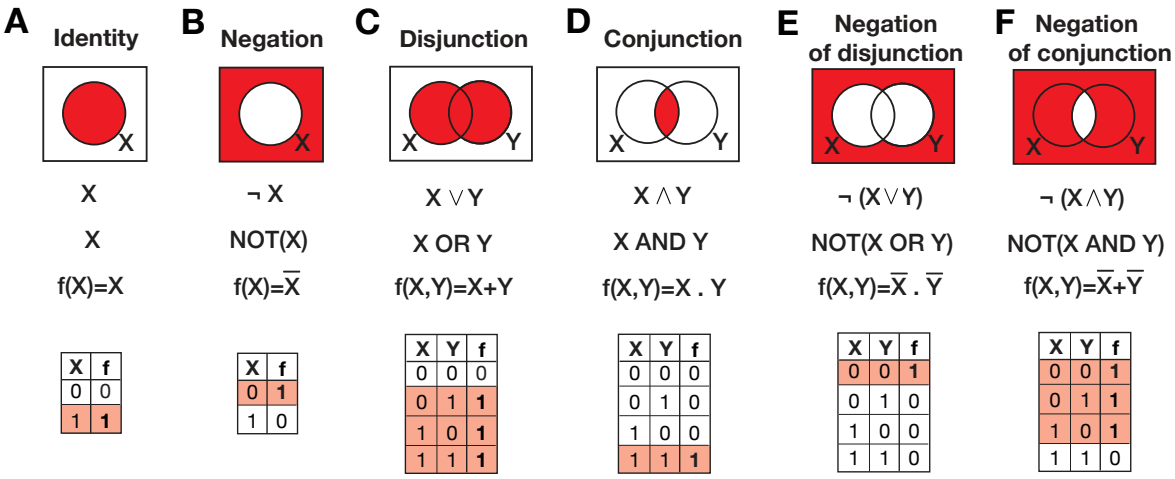


Figure 1.6: Representation of the basic logic operations: (A) identity, (B) negation, (C) disjunction, (D) conjunction, (E) negation of disjunction, and (F) negation of conjunction. For each operation, the Venn Diagram, the truth function with mathematical symbols and with common language terms, the Boolean function, and the Boolean truth table are represented. In each Venn Diagrams, the domain(s) of the graph is(are) in red when the proposition is True. The Boolean functions are written in the disjunctive normal form.

A functionally complete set is an operator set supporting the implementation of all logic functions. Logic circuits are usually built on the composition of a set of basic logic gates corresponding to a functionally complete set. The most used complete sets are:
- The two single operator sets: "NOR" (False if A or B are True, Figure 1.6D) and "NAND" (False if A and B are True, Figure 1.6E).

- OR and NOT operator set.
- AND and NOT operator set.
- AND, OR, and NOT operator set called the universal operator set.

Often for logic circuit design, a functionally complete set is chosen and logic gates implementing each operator are engineered. For implementing a Boolean function, the function will be rewritten in a minimized form using only the chosen operators. The logic gates will then be connected, usually across multiple layers (i.e. multilayering, where the outputs of some gates serve as inputs for others), to implement the desired Boolean function. The key for this design strategy lies in manipulating the Boolean function to obtain the simplest equation form that will be translated into the simplest circuit.

The "optimal circuit" characteristics are differently defined depending on the type of implementation and the application.

Generally, two parameters are taken into account with the aim to keep them to a minimum:

(1) the number of parts.
The number of parts is generally correlated with the cost of the circuit (i.e. energy consumption). Thus a reduced number of parts will decrease overall costs.

(2) the number of layers in the circuit.
The number of layers in the circuit influences the speed of the computation. Depending on the technology and application it might or might not be a limitation. Furthermore, multi-layer systems require logic gate connections, which can necessitate several rounds of optimization.

Trying to reduce the number of parts and layers, the simplest optimal circuit will be obtained from the simplest Boolean function. However, as a myriad of forms of a single Boolean function are possible, there is no completely general criterium.

Two types of circuit design can be distinguished: (1) two-layer circuits, also called second-order circuits and (2) multilayer circuits, also called factored circuits. Two-layer circuits correspond to circuits with the minimized number of layers but usually a higher number of parts than multilayer circuits. Contrarily to multilayer circuits, a minimized design for two-layer circuits can be obtained in a straightforward manner. The procedure to minimize Boolean circuits in two-layer circuits is detailed in Figure 1.7. Minimization can be performed to use either the universal operator set, only NOR, or only NAND operator sets. Unlike minimized design, multilayer circuit design is performed in a trial-and-error manner.

Simplification and minimization strategies of Boolean equations have been developed to obtain, from a truth table, the simplest Boolean equation form corresponding to a two-layer circuit. The systematic simplification is performed using NOT, AND, and OR. For circuit based on NOR or NAND operator sets, a transformation of this minimized function is then done. First, for minimization, the Boolean equation is either written as a sum of product of

literals (S-O-P), also called disjunctive normal form, or as a product of sum of literals (P-O-S), also called conjunctive normal form. The normal form corresponds to a product or sum of terms (or clauses) in which no variable appears more than once. Indeed, multiple occurrences of a variable in a sum or product term can always be simplified as it is either redundant or results in a trivial function.

A second-order Boolean function will be considered minimal if: (1) the number of terms (clauses) and (2) the number of literals in the function are minimal, with either term corresponding to a product of literals or sum of literals and separated by sum or product operations for the disjunctive normal form or conjunctive normal form, respectively.

For minimization, two different techniques are used: the Karnaugh-map (or K-map), and the Quine McCluskey method. The K-map is simple to understand and to perform by hand as it is highly visual (Figure 1.7) and the Quine McCluskey method is more fitted for automatisation as a corresponding algorithm can easily be written down.

The Karnaugh-map minimization technique is detailed in the Figure 1.7. Following a similar workflow, the disjunctive normal form and conjunctive normal form can be obtained. Therefore, the minimized form based on NOR gates is directly obtained by double negation of the disjunctive normal form. The form based on NAND gates is directly obtained by double negation of the conjunctive normal form (Figure 1.7).

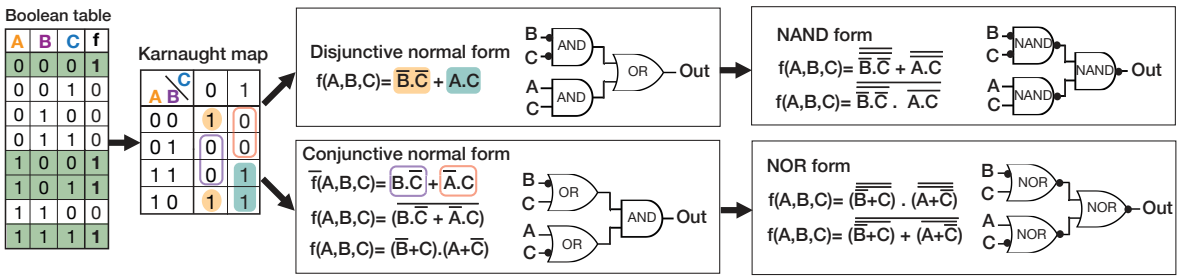


Figure 1.7: **Minimization of Boolean logic functions from the truth table to the gate diagram using Karnaugh map.** Classic Boolean truth tables are transformed in a Karnaugh map, which can be associated to a 2D truth table. Instead of having the different states of inputs in lines, states of inputs are in a table with two entries. For 3 inputs, C is 0 in the first column and 1 in the second, with lines corresponding to the different states of A and B. Then, the upper left cell corresponds to not(A).not(B).not(C) and the lower right cell to A.not(B).C. The principle is that each cell is different by only one variable to all its neighboring cells. Then, to go from K-map to the disjunctive normal form, we try to associate cells with 1 by clusters corresponding to neighbor cells. Then, each cluster can be associated to a conjunction of literals. Additional minimization steps are usually used to reduce the number of terms and the numbers of variables in each term. By double negation, the DNF (disjunctive normal form) is transformed into NAND form. Similarly, from the K-map to the conjunctive normal form, we associate cells with 0 by clusters, we obtain then the DNF of the negation of the function. Using the De Morgan's law, the conjunctive normal form of the function is obtained and, by double negation, the NOR form.

Contrary to two-layer circuit design, multilayer circuit design is not systematic due to the large number of possible rearrangement actions and solutions [Mano 2014]. Various sets of transformations are applied to Boolean functions (e.g. factoring, decomposition, extraction, substitution, and elimination) to find an optimal solution that fits with the intended specifications such as the cost, the type of parts used, and the number of layers.

In electronics, circuits are decomposed in modular parts, and this decomposition exists at different levels. At the most basic level are transistors, which are composed to form logic gates. Logic gates themselves can then be further composed in functional blocks such as multiplexers. Standard functional blocks enable the implementation of complex logic circuits with low design and optimization costs.

Here I have detailed only combinational logic circuits, but sequential circuits and analog circuits are also of great interest. Sequential circuits are circuits in which the output is dependent on the history of the system (i.e. the order of occurrence of the inputs) and not just on the current presence of inputs. For analog circuits, the inputs and outputs are not binary (e.g. 0 or 1) but can have any non-integer values.

Sequential circuits are highly used in electronics as they allow, among other things the storage of information. The two principal functional blocks in electronic circuits are flip-flops and latches. Often, electronic circuits are composed of an arrangement of sequential and combinational circuits. Sequential circuits are defined using state table and state diagrams. Unlike combinational circuits, no specific algebra has been defined to minimize sequential circuit design.

Most biological networks operate in an analog fashion; therefore, it could then be of interest to implement analog genetic circuits instead of digital circuits. However, digital circuits are more suited to perform computation. Indeed, in electronics, computation is performed using digital circuits and converters are used whenever one uses an analog signal as input (A-D converter) or desires an analog signal as output (D-A converter).

1.2.1.3 Biological systems as computational systems

Many scientists took inspiration from living systems to design complex artificial automata, such as cellular automata. An automaton is a machine that operates on its own without human control. For decades, scientists viewed cell as natural automata performing complex computation. Their objectives were both to model cellular mechanisms by defining logic rules of behavior and to construct artificial automata inspired from the complexity of natural automata.

In 1943, McCulloch and Pitts worked on the logical calculus inspired from human nervous activity [McCulloch 1943]. Based on the properties of neurons that were known at this time, they proposed a simple model called the “formal neuron operations”, in which the neuron receives several inputs and sum them as an output. These formal neurons are nowadays the elementary units of artificial neuron networks used for artificial intelligence. However, formal

neuron behavior is too simple in comparison to the behavior of real neurons, and so cannot be used for the deep understanding of nervous systems.

During the last years of his life, John Von Neumann worked on the Theory of Self-Reproducing Automata [Von Neumann 1996]. He compared natural and artificial automata and highlighted the computing efficiency of natural automata that was at the time thousands of times higher than that of artificial automata. Among others, he highlighted the flexibility, autonomy, and self-reorganisation of cellular systems that allow them to survive even in high incidence of error.

In Von Neumann's mind, the paramount biological phenomenon was the self-reproduction of living organisms. The self-reproduction permits demultiplication of components, which by interaction leads to complex systems (i.e. multicellular organisms). Von Neumann highlighted the fact that in nature, self-reproduction leads to an increase of complexity of organisms. As organisms are error-tolerant, non-lethal and non-deleterious mutations are inherited and can lead to an increase in organism complexity. Based on these observations, Von Neumann formulated a fundamental question: "What kind of logical organization is sufficient for an automaton to be able to reproduce itself?". As an answer, he imagined a self-replicating cellular automaton [Mitchell 1998]. This idea of cellular automaton has been further developed and is now a well-known model studied in a large number of fields such as computer science, mathematics, and theoretical biology.

The view of living organisms as automata defined by a limited set of axioms was followed by scientists who attempted to model cellular behavior, culminating in the field of systems biology. By extension, natural protein networks and gene regulation pathways are seen as logic circuits [Arkin 1994, Hjelmfelt 1991, Hjelmfelt 1993] [Bray 1995]. Based on this idea of natural systems behaving as logic circuits, various synthetic logic circuits have been and are implemented using biological components, *in vitro* or *in vivo*.

1.2.2 *In vitro* biocomputing

In 1994, the computer scientist Leonard Max Adleman used DNA as a computational system [Adleman 1994] to solve the Hamiltonian path problem: an NP-complete problem (problem requiring a time exponential to the size of the input data to be solved). Adleman's DNA computation was based on the Watson-Crick complementarity of DNA. He used 20 base pair oligonucleotides for each vertex and edge of the path and ligation/PCR rounds to obtain the result of computation. While the solution to this seven-node Hamiltonian path is trivial, it was the first successful computation using DNA and Adleman suggested that the method could be scaled-up to much larger graphs. This paper is widely regarded as the beginning of the field of DNA computing.

The speed of any computation, biological computation or others, is determined by two

factors: "(i) how many parallel processes it has and (ii) how many steps each can perform per unit time." [Lipton 1995]. Biological computation is slower than in electronics but it can be highly parallelized. Indeed, as Lipton stated: "As little as 3 g of water contains approximately 10^{22} molecules" [Lipton 1995]. Lipton used the same principle as Adleman to solve the SAT problem (or Boolean satisfiability problem).

For these two resolutions of computational problems, the inputs of the computation are oligonucleotides, the output is a specific DNA sequence, and the computation is performed in test tubes. Many studies were derived from these first two papers. The field of *in vitro* DNA computation has since split in two realms: computation based on Watson-Crick complementarity and computation based on enzymatic reactions driven by ribozymes and deoxyribozymes.

Ribozyme-computing

Ribozymes are RNA molecules that catalyse a specific enzymatic reaction. Their role is essential in key biological processes such as translation, RNA splicing, or viral replication [Guerrier-Takada 1983, Kruger 1982]. Deoxyribozymes, similar to ribozymes but based on DNA, are not found in nature but have been engineered [Breaker 1994].

Ribozymes and deoxyribozymes have been used to implement Boolean logic gates. Stojanovic [Stojanovic 2002] engineered a set of deoxyribozyme-based 2-input Boolean logic gates (NOT, AND, and XOR logic gates). As the inputs and outputs of these gates are oligonucleotides, these logic gates can be connected to implement more complex functions. Consequently, all Boolean functions are theoretically implementable by layering NOT, AND and XOR gates, from this complete Boolean set. The layering of simple logic gates here permits the implementation of much complex circuits than with previously described DNA-based systems.

This method was applied to build a Tic-Tac-Toe automaton [Stojanovic 2003]. To do so, NOT, AND, AND/AND, and AND/AND/NOT deoxyribozyme logic gates were used. Various deoxyribozymes were placed in different compartments and an OR function was performed by using a deoxyribozyme cleaving the same substrate. The automaton never lost against a human, as a prefect strategy was implemented. The automaton play was shown by fluorescence. Using the same strategy, a full binary adder was engineered [Lederman 2006]. Limitations of the deoxyribozyme-based logic circuit are: (1) it only takes single stranded DNA as an input, which reduces the potential range of applications; and (2) logic gates with more than three inputs have not yet been shown and might be challenging to engineer.

Computing using Watson-Crick pairing and DNA strand displacements

In addition to the first two fundamental papers, many of studies have used the Watson-Crick complementarity property of DNA for computation [Padirac 2013], such as toehold-based reaction circuits. The principle of a toehold-based circuit is to use a small single-stranded recognition sequence (toehold) to control the displacement of an "output" strand by an invading "input" strand [Yurke 2000]. This simple approach permitted the development of a full set of

Boolean logic gates [Seelig 2006]. The simple logic gates were difficult to connect, limiting the expansion of toehold-based computation circuits. Qian and Winfree then developed the seesaw gate: a compact gate motif which allows gate layering. Qian and Winfree used this technology to construct two large computation circuits, one which calculates the square root of a four-bit binary number ([Qian 2011a] and one that mimics neural network computation [Qian 2011b].

Toehold-based reactions are irreversible. To obtain a reversible reaction, a continuous source of energy is needed: this was achieved using enzymatic reactions. Two systems were developed using a similar principle: Genetlet and DNA-toolbox.

(1) Genetlet is based on RNA transcripts that regulate their own transcription from DNA gene analogs [Kim 2006]. RNA is both the input and output of a circuit with a DNA-encoded software and an enzymatic hardware composed of RNA polymerase and RNase. Genetlet technology supported the implementation of a bistable switch [Subsoontorn 2012b] and of various oscillators [Kim 2011].

(2) The DNA-toolbox [Montagne 2011] is based on two types of DNA signals: input DNA activating DNA templates and inhibitors blocking DNA templates. In both cases, an exonuclease degrades the signal molecules and not the DNA templates. Using the DNA-toolbox, various complex circuits were implemented such as an oscillator, a bistable system, a switchable memory [Padirac 2012], and a reaction-diffusion French-Flag pattern [Zadorin 2017].

The recent development of *in vitro* DNA-based computation toolboxes used for diagnostic application to detect specific RNA or DNA sequences is of interest [Pardee 2016]. DNA is highly flexible, adaptable and, unlike enzymes, exhibits simple-to-predict behavior.

Computing with enzymatic reactions

Using only enzymes, complex computation systems can theoretically be implemented using a network of enzymatic reactions. A theoretical design of a neural network and a Turing system were described by Hjelmfelt and colleagues as early as 1991 [Hjelmfelt 1991, Hjelmfelt 1992]. Such *in vitro* enzyme-based circuits have not yet been implemented. Indeed, the complexity of the implementation rests in the lack of enzymes performing the theoretical reactions with full orthogonality. Additionally, the design of enzymes performing specific reactions is still a challenging task.

Using a natural enzymatic system, several 2-input logic gates were implemented using maltose, phosphate, and sucrose as inputs [Zhou 2009][Privman 2010]. In addition to logic gates, the authors constructed an amplification system and conversion system transforming outputs into inputs to permit connection of logic gates. Consequently, using compartmentalization in a microfluidic channel, these tools could be used to implement a multi-layer logic circuit. However, this system is not modular and cannot accept different inputs as it is based on specific enzymes responding to specific inputs. New circuits based on the same principle could be engineered but would need a full new round of optimization.

Following this idea of compartmentaling of enzymatic circuits, Courbet and colleagues [Courbet 2015b, Courbet 2018] implemented standard logical operations by biochemical networks encapsulated and insulated within synthetic vesicles called protocells.

This last work shows the power of compartmentalization in biocomputing. In electronic circuits, components are physically separated and connected specifically as desired using wires. An identical component can be reused and wiring is performed to obtain the desired circuits. In biological systems, all components are in the same compartment and can interfere with each other. Consequently, various orthogonal components are needed when used in the same compartment (e.g. test tube, single cell). However, if the system requires "chemical wires", i.e. molecular communications channels to establish connections between subcompartments, the number of orthogonal channels is a limiting factor.

1.2.3 Implementing logic circuits in living organisms - *in vivo* bio-computation

Using *in vitro* bio-computation, the exact composition of the system is known and controlled, which simplifies predictions of system's behavior. On the other hand, the implementation of logic within living organisms allows us to build upon complex functions already performed by living organisms. Many computing circuits operating within living organisms have been built that hijack or rewire natural regulatory mechanisms.

Gene expression is a common and tractable output in natural biological systems, and genes are generally expressed in response to specific signals or signal combinations. Moreover, many mechanisms regulating gene expression are relatively well understood and amenable to being more easily engineered. Consequently, *in vivo* bio-computations are based on the regulation of gene expression.

The implementation of large logic circuits *in vivo* requires: (1) a library of orthogonal parts performing a complete functional operator set and (2) the connection of these parts in a straightforward manner to implement larger circuits. *In vivo*, most signals are analog and digitization of the input signal is required to obtain digital logic. A compact design (i.e. a reduced number of parts) is prefered to simplify the construction and optimization of the circuit and, more importantly, to not overload cellular metabolism.

As the implementation of computation in living organism is the subject of interest of my thesis, I will introduce existing circuits in more details in an overview of the different kind of mechanisms that have been used to compute within cells.

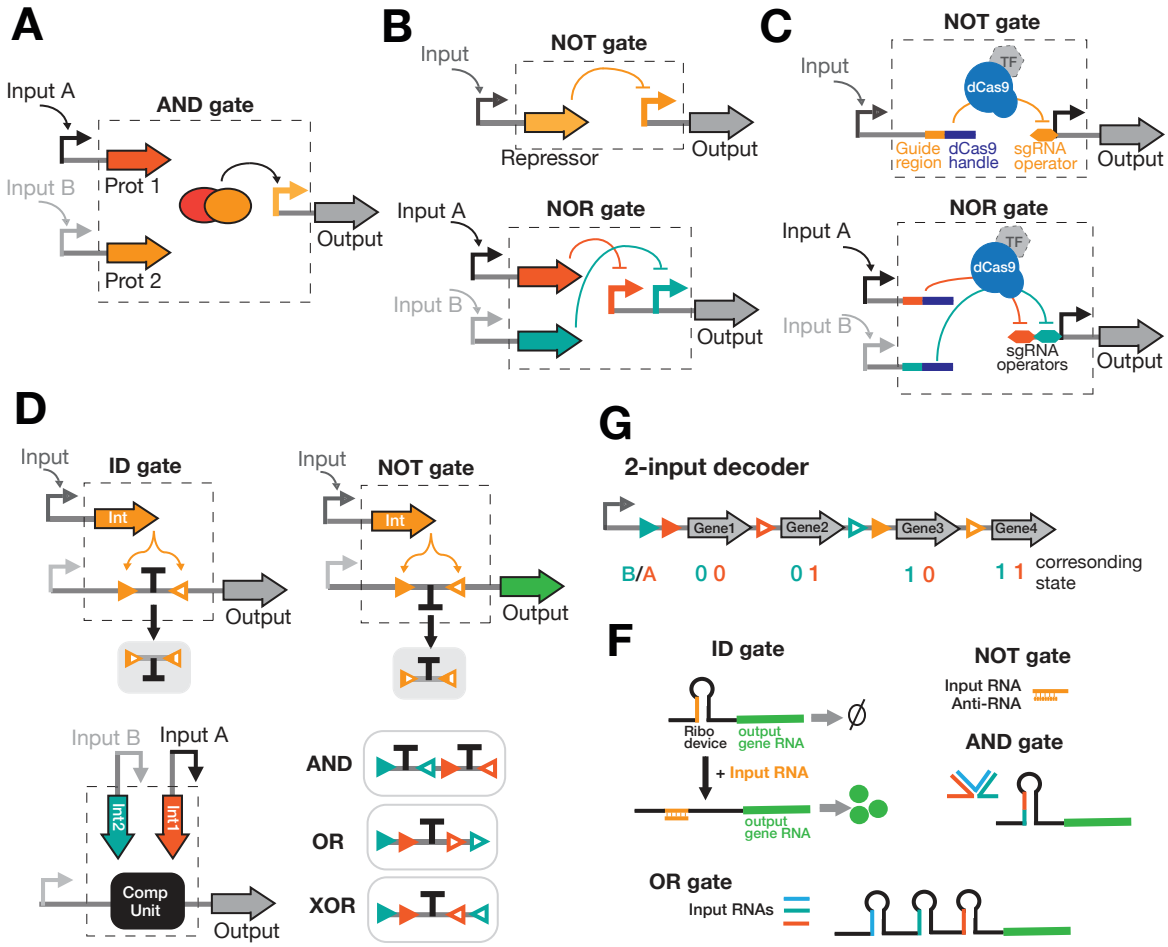


Figure 1.8: **Tools for implementing of logic in living organisms. A - 2-input AND gate based on two proteins required for activation of transcription.** Each input induces the expression of one protein, where in presence of the two proteins the output promoter is activated. This principle has been applied with the hrpS-hrpR regulator of polymerase [Wang 2011], chaperone-transcription factor [Moon 2012], and split T7 RNA polymerase [Shis 2013]. **B - Repressor-based NOT and NOR gates.** The input induces the expression of a repressor, which represses the output gene, forming an NOT gate. By placing two operator sites specific to two repressors in tandem, if one input is ON, a repressor is expressed, which inactivates the promoter and the output. Nielsen et al., constructed repressor based gates based on a library of Tet-family repressors [Stanton 2014, Nielsen 2016]. **C - CRISPR-based NOT and NOR gates.** dCas9 has been engineered to repress transcription in response to sgRNAs that target specific operators. Depending on the organism in which this technique is applied, the dCas9 is fused to a specific transcription regulator, either a repressor or an activator. Gander et al., used this technique in yeast, fusing dCas9 to a repressor [Gander 2017]. Then, inputs induce expression of sgRNA that repress expression of the output gene via dCas9, generating a NOT gate. For NOR gates, two sgRNA operators are placed in series in the promoter site, then in presence of one input, the output promoter is repressed.

Figure 1.8: **D - Gates based on recombinase terminator switches.** Serine integrases recognize specific integrase site pairs and invert or excise DNA between sites depending on site orientation. Bonnet et al. designed simple transcriptional switches placing integrase sites around asymmetric terminator [Bonnet 2013]. For the ID-gate, a terminator blocks transcription of the output gene, and in presence of the input the terminator is inverted and transcription occurs. For the NOT-gate, the terminator is in an inverse orientation than for the ID-gate, where the output gene is transcribed only in absence of the input. Different computing units composed of integrase sites and terminators have been engineered to implement all 2-input logic functions. **G - Excision-based logic gates.** Still using integrase, Weinberg et al., built a logic scaffold based on excision and integrase site variants permitting implementation of all logic circuits. For implementation of specific logic functions, for the ON state in the truth table, the output gene is placed at the corresponding location in the 2-input decoder. **F - Riboregulator-based logic gates.** A RNA stem-loop placed upstream a gene sequence inhibits ribosome binding and translation of an output gene. Unfolding of the stem-loop is mediated by an input RNA which then activates translation of the output gene, leading to an IDENTITY gate [Isaacs 2004]. Green et al. extended this system [Green 2017] using an additional RNA inhibiting the input RNA which permits implementation of the NOT function by inhibiting the unfolding of the stem loop. Then, an AND gate is formed by cooperativity between input RNA and several input RNA are required to form the stem-loop inactivating taRNA. Finally, riboregulators are placed in series, where unfolding of one stem-loop leads to activation of the output gene, forming an OR gate.

1.2.3.1 Computation based on transcription regulators

Many computation circuits have been implemented using transcription regulators. Jacob and Monod [Jacob 1961] discovered the regulation mechanism of the Lac operon and its transcription regulator, the Lac repressor LacI. A transcription regulator activates or represses transcription by binding to specific DNA sequences positioned within or around the promoter region. The activity of transcription regulators is usually controlled by specific molecules. For example, the Tet repressor prevents transcription at the Ptet promoter. Binding of tetracycline or anhydrotetracycline to the repressor triggers a conformational change leading to its release from the promoter and subsequent transcriptional activation. Natural transcription regulators such as LacI, TetR, cI, cAMP, and AraC provided a toolbox for implementing computation based on DNA transcription. In 1994, Joung and colleagues [Joung 1994] pioneered the engineering of an artificial promoter bearing binding sites for two different regulators, cI and cAMP receptor proteins. This artificial promoter behaves similarly to an AND gate, as the presence of both proteins lead to an induction larger than the addition of each protein alone.

In 2000, two genetic circuits using the repressors LacI, TetR and cI arranged into mutual feedback loops were engineered for the implementation of a toggle switch [Gardner 2000] and a

repressilator [Elowitz 2000]. Moreover, a library of circuits, some of them having binary, logic responses was generated by combining parts from the repressilator in a combinatorial manner [Guet 2002]. Kramer and colleagues [Kramer 2004] constructed artificial promoters with up to three operator sites specific for mammalian transcriptional regulators. Coupled with the parallel and serial linking of two-gene regulation systems, they engineered "BioLogic"gates able to respond to up to three inputs in mammalian cells. However, their design was not modular enough to implement all 2-input logic circuits.

In addition, various types of transcription factor or polymerase activation methods have been engineered. Several AND gate designs are based on the requirement of two or more proteins for promoter activation, such as the amber suppressor tRNA supD and T7 RNA polymerase with two amber strop codons [Anderson 2007], the hrpS-hrpR regulator of polymerase [Wang 2011], three orthogonal chaperon-transcription factors [Moon 2012], and a split T7 RNA polymerase [Shis 2013] (Figure1.8A). Additionally, a sigma/anti-sigma library was built to allow the construction of several orthogonal IDENTITY and N-IMPLY gates [Rhodius 2013].

Later, Stanton and colleagues [Stanton 2014] developed a library of 16 orthogonal repressors based on genome mining of TetR-family repressors. Using this repressor library, they constructed various NOT and NOR gates. NOT gates (also called "inverters") place a repressor under the control of the signal and the output under the control of this repressor. The resulting circuit is ON when the input is OFF and vice-versa. NOR gates are designed using tandem promoters composed of two repressor-operator sites. When at least one input is present, the transcription is repressed (Figure 1.8B). Simple gate multilayering was possible by expressing repressors as inputs for the downstream gate. By wiring this large set of NOR gates, most of the 3-input logic gates were implemented in single cell in *E. coli* [Nielsen 2016].

In parallel, researchers designed Distributed Multicellular Computation systems in which logic circuits are divided into different strains within a multicellular system [Tamsir 2011] [Regot 2010] [Macia 2014] [Urrios 2016] [Macia 2016]. Regot and colleagues constructed a library of 16 *S. cerevisiae* strains implementing NOT and IDENTITY gates that were used in a multicellular system via cell-cell communication to compute up to 6-input logic functions. More details on the designs, limitations and advantages of these two single-cell and multicell logic implementations can be found in the following subsection 1.2.2.

Other systems regulating transcription were used to implement logic, including zinc fingers [Lohmueller 2012], TALEs (transcription activator-like effectors) [Lienert 2013] [Gaber 2014] and CRISPR-dCas9 [Nielsen 2014, Kiani 2014, Gander 2017]). These systems can operate in eukaryotes and large libraries of orthogonal components can easily be engineered. More details on these circuits can be found in Annex B.

Using large orthogonal libraries of repressors and CRISPR-dCas9-sgRNA, large logic circuits are theoretically implementable by layering NOT and NOR gates. The main limitation of these circuits is the requirement of multiple layering of these simple 2-input gates. For multicellular

systems, the limiting factor is the number of cell-cell communication channels. The computation performed using these systems is real time in contrast to systems based on DNA recombination that I will describe.

1.2.3.2 Computation based on regulation of translation

A large variety of engineered non-coding RNA toolboxes with distinct functions exists, for more details see the review of Qi and Arkin [Qi 2014]. As an example, riboregulators permit regulation of translation by triggering the unfolding of a stem loop structure in the mRNA, exposing the RBS for ribosome access. MicroRNA (miRNAs) target mRNAs for degradation and pT181-RNAI-type elements bind 5'UTR elements triggering the formation of premature transcriptional terminators [Lucks 2011].

These large toolboxes have been used to implement computation circuits [Benenson 2009]. For the detection of endogenous mRNA, logic circuits were built in human cells based on siRNA repressing the expression of the output. By coupling siRNA-based circuits with transcription activator and repressor, up to 5-input logic circuits were built [Rinaudo 2007] [Xie 2011].

In addition, riboregulator-based circuits were implemented in living organisms. The systematic engineering of riboregulators enables the specific translational control of gene expression [Isaacs 2004]. This mechanism was used by Green and colleagues to implement up to 5-input logic circuits [Green 2017] (Figure 1.8F). Various riboregulators sensing specific taRNA were designed *in silico*. For the implementation of an IDENTITY gate, the input RNA unfolds the stem-loop which activates translation. For NOT gates, the input RNA inhibits another RNA inhibiting the activation of translation mediated by this secondary RNA. For AND operators, input RNAs cooperatively activate translation. Finally, an OR operator is performed by placing riboregulators in series, which allows independent induction of the translation by each riboregulator.

The previous systems are induced by RNA, but riboregulators controlled by ligands were also engineered in mammalian cells [Bayer 2005]. Riboregulators controlled respectively by theophylline and tetracycline were engineered for inhibition and activation of translation. While the previous system is based on a stem-loop, a similar system was engineered based on ribozymes, such as self-cleavage of RNA induced or repressed by theophylline and tetracycline [Win 2007]. By combination of an aptamer binding to theophylline and tetracycline in a different position of the ribozyme, AND, OR, NAND, and NOR gates were engineered [Win 2008]. To scale up these circuits, ribozymes with effector binding sites responding to other molecules are needed, but their design is not an easy task [Townshend 2015].

Systems using RNA as input to control translation are relatively scalable. However, the sensing of other types of inputs by RNA devices is still limited.

1.2.3.3 Protein based computation

In vivo computation can also be performed using enzyme networks such as *in vitro* systems. *In vivo* enzyme networks permit to use the natural cellular networks as template for engineering complex behavior. Dueber and colleagues [Dueber 2003] reprogrammed the control of an allosteric signaling switch : the actin regulatory protein n-WASP (neuronal Wiskott-Aldrich syndrome protein). The activity of the protein is naturally repressed by autoinhibitory interactions involving two domains: the GTPase-binding domain (GBD) and a basic (B) motif. Two inputs induce the activity of the protein: GTP-loaded Cdc42 for GBD and PIP2 for the B motif inactivate the autoinhibitory interactions. By generating various combinations of protein domains, different logic systems were generated such as: a single input response, an AND gate, and an OR gate. A chemically induced dimerization (CID) system was used to engineer simple logic gates in mammalian cells. Miyamoto and colleagues used two orthogonal dimerization systems, GA3-AM and rapamycin to implement OR and AND gates. Using this system, the output is obtained in a timescale of seconds, which is faster than other *in vivo* circuits, such as the one of Duber and colleagues.

All actual enzymatic-based systems are highly specific and are consequently not adaptable to various input molecules. However, fast logic circuits, as shown by Miyamoto and colleagues [Miyamoto 2012], are only obtainable using logic circuits that do not required neither transcription nor traduction to produce a response.

1.2.3.4 Computation based on DNA recombination

Computation based on DNA recombination uses recombinases as a tool to modify DNA in a heritable manner. Recombinases are naturally used for DNA manipulation, for example by phages to integrate their genomes into the bacterial genome. Recombinases families, their mechanisms, and their applications to design logic circuits are detailed Section 3 of this chapter. I will provide here a brief overview of the main recombinase logic circuits.

The largest logic circuits implemented with recombinases are based on the serine integrase sub-family. Indeed, serine-integrases mediate a precise, efficient, and irreversible recombination of DNA without requirement of co-factors. It is possible to produce integrase-mediated DNA inversion or excision by orienting the integrase sites in either parallel or antiparallel orientations.

2-input asynchronous Boolean logic gates were implemented in a single layer using serine integrases [Bonnet 2013, Siuti 2013] (Figure 1.8D). These logic circuits are implemented by placing promoters, terminators and genes between integrase sites. Using only serine integrases, the DNA switch is irreversible, and thus the logic implemented is asynchronous. However, reversible switches were implemented using RDF-integrase circuits [Bonnet 2012, Subsoontorn 2014] and could be used for implementation of reversible integrase-based logic gates.

Based on a different design strategy, Weinberg and colleagues [Weinberg 2017] engineered integrase-based asynchronous Boolean logic in mammalian cells. Their system is based on DNA excision exclusively and uses not only serine integrases but also tyrosine integrases. Weinberg and colleagues implemented circuits responding to 3 inputs.

In addition to Boolean logic, history-dependent logic is implementable using the irreversibility property of serine integrases. Hsiao and colleagues [Hsiao 2016] implemented a 2-input temporal logic gate permitting the differentiation of the order of occurrence of 2 inputs. Repetition of these modules was used to record all possible 3-input sequential states [Roquet 2016]. Based on this recorder design, various 2- and 3-input history-dependent gene-expression programs were implemented.

One advantage of integrase circuits is to support the implementation of complex functions in a compacted circuit. As I worked during my thesis on the implementation of logic in living organisms using integrases, I will detail the mechanism and the various usages of recombinases in Section 1.3.

1.2.4 A comparison of the different design strategies for *in vivo* implementation of Boolean functions

I presented the different logic circuits built using biological components. Various Boolean logic circuits have been implemented in living organisms using different biological components and on various design strategies.

The choice of the design strategy can be separated in two steps: (1) the definition of building block, such as simple gates, and (2) the determination of the type of connection between building blocks. The definition of the building blocks mainly depends on the type of biological components used. As shown in Figure 1.8, some components permit the implementation of NOT and NOR gates within a single layer (e.g. repressors), and others permit implementation of a more various range of gates within a single layer (e.g. recombinases). One of the keys permitting scalable circuits is automation of the design, especially for an increasing number of inputs.

I will compare here two multilayer design strategies for which an automated design workflow exist, one single-cell and one multicellular design.

Based on transcription factors, most designs are based on the layering of one or two input logic gates to implement complex Boolean functions. Nielsen and colleagues and Macia and colleagues succeeded to develop a workflow for the systematic implementation of large Boolean functions based on this layering design, either in single cell [Nielsen 2016] or in multi-cell via cell-cell communication [Macia 2016].

1.2.4.1 Automated design of single-cell logic circuits: Cello

Here, I will focus on Nielsen and colleagues circuit design based on repressors. It is currently the largest set of logic circuits implemented in living organisms [Nielsen 2016]. Their system is based on the layering of 2-input NOR gates and single-input NOT gates. Boolean functions are then implemented in multilayers, corresponding to a factored form of the Boolean function (Section 1.2.1). Consequently, the simplification to the reduced circuit design is not straightforward. Nielsen and colleagues developed a design environment called Cello, which automatically transforms logic functions into DNA sequences. The Boolean function simplification is performed via Cello. First, a synthesis tool, ABC ,generates from the truth table an AND-Inverter Graph (composed of 2-input AND and NOR gates) and minimizes the number of gates and layers. This graph is converted using DeMorgan's rules in a NOR-inverter graph. The result of the ABC algorithm is not necessarily the simplest solution. To obtain the simplest circuits, logic motifs are switched to equivalent subcircuits via a brute force method generating all possible circuits.

To generate the DNA sequence fitting to the input Boolean function, a User Constraint File containing all NOT and NOR gate behavior is used to predict the propagation of the signal in the circuits. It permits the creation of a model of the circuit behavior. As the number of possible circuits is too large, a search algorithm was designed to perform this task.

Cello is essential to design repressors-based single-cell logic circuits, as Boolean function minimization is not straightforward, and neither are gate connections. Circuits implemented using new inputs and in a new organisms will required precise characterization of the new parts to be adaptable to Cello. Despite the minimization process, repressor-based circuits require a large set of parts and of layers increasing the metabolic load to cells and the computation times. Due to the limitation of the characterized repressor parts used for model prediction, some logic circuits were scored as impossible via Cello and have not been constructed. To scale-up the circuit design, as some 3-input logic circuits are composed of up to 50 parts, it seems difficult to envisage 5- or 6-input logic circuits using this technique. However, efforts are underway to optimize this workflow and permit implementation of larger circuits.

Cello is the only design environment suited for logic gate design in living organisms. Its capacity has been shown for repressor-based circuits, but could be adapted to others tools.

1.2.4.2 Distribution of computation in multicellular systems

Macia, Solé, Posas, and colleagues used in several papers the distribution of computation in multicellular consortia for the implementation of complex logic circuits in *S. cerevisiae* [Regot 2010] [Macia 2014] [Urrios 2016] [Macia 2016].

Macia and colleagues based their design on the layering of NOT and IDENTITY gates

implemented in separated strains [Macia 2016]. They used one communication channel and physical separation. First, a set of 16 *S. cerevisiae* strains computing either NOT or IDENTITY gates were constructed and characterized. In this set of strains, input layer cells detect the input signal and produce as output: the pheromone used for communication. Output layer cells, composed of a NOT gates, detect the pheromone and produce as output: the final output signal of the system.

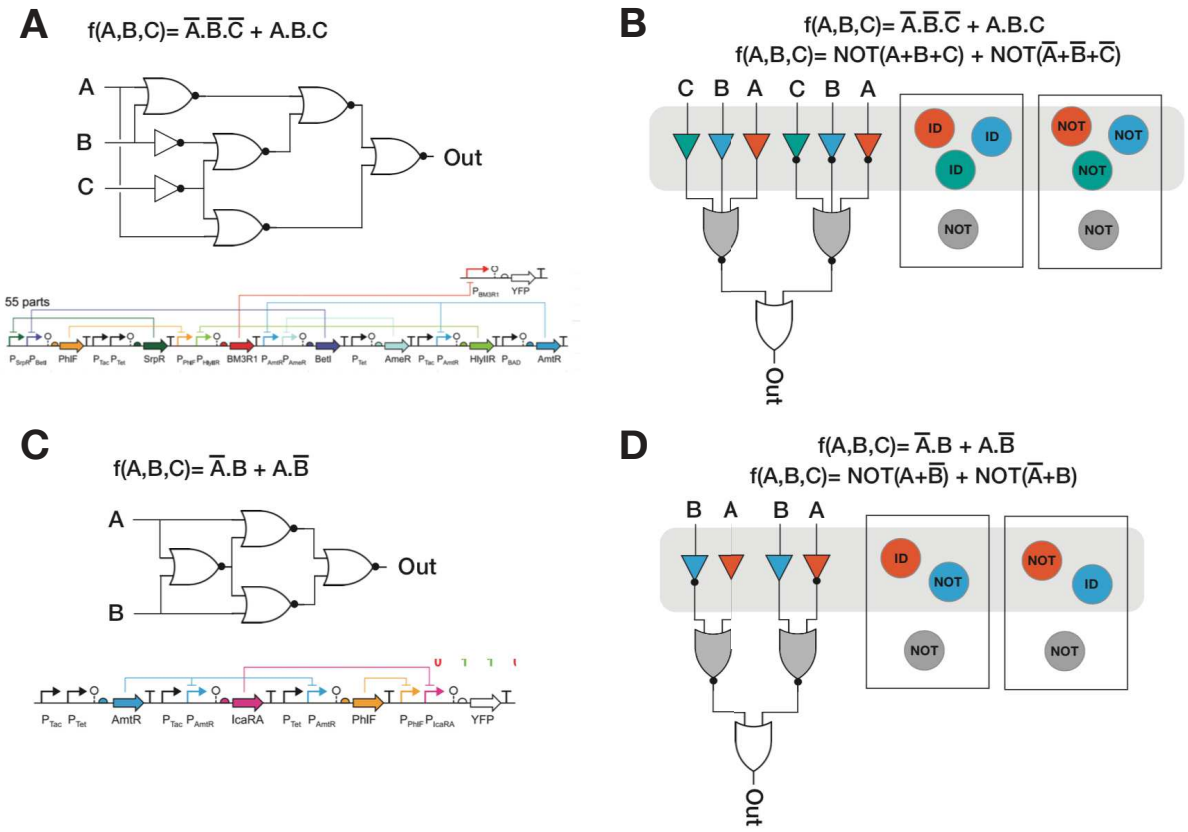


Figure 1.9: **Multicell and single cell implementation of Boolean logic functions.** (A) and (C) are the designs from [Nielsen 2016] and (B) and (D) are the designs from [Macia 2016]. (A)-(B) corresponds to the implementation of the logic function not(A).not(B).not(C)+A.B.C and (C)-(D) corresponds to the implementation of the logic function no(A).B+A.not(B) (XOR gate).

They defined a systematic method of design for implementation of all Boolean functions. First, the Boolean function is written in the DNF (disjunctive normal form) and transformed as a disjunction of negative disjunction of literals using the De Morgan's law (corresponding to OR of NOR of NOT and ID functions). Negation and identity of input signal is performed using input layer cells. The negative disjunction is computed by the output layer cell, if one of the input layer cells expresses the pheromone (is ON), the output layer cell will turn OFF. Each clause of the disjunction is physically separated; then, if in one of physical compartment the output cell is ON, the output of the computation is considered ON, computing then the full function. Optionally, an additional layer is added to permit integration of the output signal,

using a buffer cell and another compartment placed downstream.

This simple and straightforward design was used to implement a 6-input multiplexer (MUX 4-to-1). Implementation of large circuits only require the composition of the set of strains, which consequently permit without effort implementation of large and various circuits. New input layer cells will have to be engineered to compute new inputs; however, circuit implemented in each cell is relatively simple and required a small number of parts. Furthemore, the characterization of a reduced number of components permits implementation of a large set of circuits. The limitation of this design is the requirement of spatial separation and consequently of human intervention, which might limit the range of applications.

1.2.4.3 Comparison of single-cell vs multi-cell designs

These two works proposed advanced and complementary framework for the design and implementation of logic circuits in living organisms.

Using distribution of the computation in multicellular consortia, a reduced number of cells can be constructed and characterized and allow by combination the implementation of a large number of complex logic circuits. Comparing to single-cell implementation, a lower number of orthogonal components is required. Additionally, a reduced number of components (strains) have to be constructed and engineered for the implementation of a large set of Boolean functions. However, this multicellular design requires the use of a physical separation or multiple cell-cell communication channels. While the single-cell implementation requires the engineering of large genetic circuits, it permits cellular computation without physical intervention of humans and can be used for the detection of various cellular patterns in living organisms.

For the design framework, the single-cell implementation required the use of a complex algorithm for the minimization of the number of layers and components required in the circuits while the multicell design framework is straightforward as no minimization is performed.

To conclude, these two designs are complementary and both strategies have their advantages and drawbacks. During my thesis, I implemented logic circuits in both multicellular and single cell systems.

1.3 Recombinases: tools for DNA editing

1.3.1 Serine and tyrosine recombinases and their mechanisms

1.3.1.1 Global mechanism of site-specific recombinases

Site-specific recombinases are enzymes that trigger a recombination process involving reciprocal exchange between specific DNA sites. Globally, site-specific recombinases recognize two specific DNA sites, break and rejoin the DNA without use of important energy and with DNA conservation. Recombinases are separated in two different families based on the amino acid involved in the DNA breaking, such as serine or tyrosine.

Depending on the arrangement of the recombination sites, recombinases mediate integration, excision, or inversion of DNA between the recombination sites (Figure 1.10). For integration, the two sites are positioned on two different DNA molecules, one of which must be circular. When recombination sites are located within the same DNA molecule, the mechanism is dependent on the relative site orientations. Sites in the same orientation (head-to-tail sites) result in an excision while inversion is performed with sites in opposite orientation (head-to-head sites).

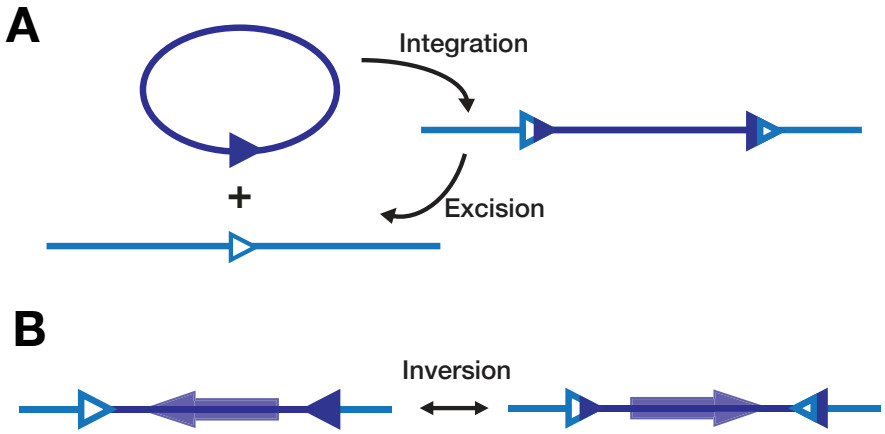


Figure 1.10: Possible recombination outcomes. A - DNA integration and excision. For integration, two recombination sites have to be on two different DNA molecules, one of which must be circular. Excision of the DNA between recombination sites occurs when the sites are in parallel orientation. B - Inversion. The inversion of DNA between the recombination sites occurs when the sites are in antiparallel orientation.

Consequently, this mechanism has a variety of biological functions in different organisms, such as integration of the bacteriophage genome in the bacterial chromosome, inversion to switch gene expression, and reduction of DNA dimers. The most studied recombinases are: the Lambda Int families mediating the integration of the bacteriophage Lambda into the *E. coli* chromosome [Nash 1981], the Tn3 resolvase mediating the resolution of cointegrates from transposition of Tn3 transposons [Stark 1989], and the Hin recombinase mediating DNA inversions for flagellar

phase variation in *Salmonella* [Feng 1994]. A more detailed list of recombinases and their biological functions can be found in Figure 1.11 or [Grindley 2006]).

Table 1 Site-specific recombination: a sampling of enzymes and functions

Recombinase	Biological function
Tyrosine recombinase family	
λ Int and many other phage integrases	Integration and excision of phage genomes
Int of Tn916/Tn1545	Integration and excision: transposition of circular transposons
IntI	Integration and excision of gene cassettes in integrons
Cre	Excision: dimer reduction in phage P1 plasmids
XerC/D	Excision: dimer reduction in the *E. coli* chromosome as well as in many other bacterial chromosomes and some plasmids
TnpI of Tn4430	Excision: resolution of cointegrates resulting from transposition of Tn4430
FimB, FimE	Inversion: alternation of gene expression (fimbrial phase variation in *E. coli*)
Rci of R64	Inversion of shufflon segments in plasmid R64, producing various forms of pili
XisA, XisC	Excision: for developmentally regulated gene activation in *Anabaena*
Flp	Inversion: for amplification of yeast 2-μm plasmid
Serine recombinase family	
TnpR of Tn3/γδ and related transposons	Excision: resolution of cointegrates resulting from transposition
Sin of *Staphylococcus aureus*	Excision: dimer reduction in staphylococcal plasmids
ParA of RP4	Excision: dimer reduction in plasmid RP4
Hin	Inversion: alternation of gene expression (flagellar phase variation) in *Salmonella*
Gin, Cin	Inversion: alternation of gene expression (tail fiber proteins) in phages Mu and P1
OrfA of IS607/IS1535	Integration and excision: transposition of the *Helicobacter pylori* element IS607 (and others?)
Int of ϕC31/Bbv1/ϕRv1[a]	Integration and excision of *Streptomyces* and mycobacterial phages
TnpX of Tn4451[a]	Integration and excision: transposition of Tn4451 in *Clostridium*
SpoIVCA (CisA)[a]	Excision: for developmentally regulated gene activation in *Bacillus subtilis*
XisF[a]	Excision: for developmentally regulated gene activation in *Anabaena*

[a]Members of the large serine recombinase subfamily.

Figure 1.11: Site-specific recombination: a sampling of enzymes and functions [Grindley 2006]
.

The site-specific recombination process can be divided into several simple steps. First, the recombinase binds as a dimer to the two recombination sites forming a synaptic complex with the juxtaposed sites. Then, the recombinase mediates the cleavage, strand exchange, and rejoining of the DNA. Finally, the recombined DNA is released via the breaking down of the synaptic complex.

The simplest recombination sites are around 20-30 bp and are composed of a pair of recognition sites binding to one dimer or two monomers of the recombinase. A DNA break occurs between the two recognition sites in the crossover site. For many recombinases, sites are 100 bp or more, as they are composed of additional sites for protein recognition. Indeed, recombination may involves the binding of several recombinases and the recruitment of co-factors.

Recombination is performed via a tetrameric complex. In each recombination site, two enzymes bind forming a dimer. The two sites are juxtaposed to form a synaptic complex. The DNA is then broken at the crossover site via either a tyrosine or serine amino acid. The amino acid acts as a nucleophilic attack on the phosphate of the DNA backbone permitting the break of the DNA and the creation of either a 3' phosphotyrosine or a 5' phosphoserine

linkage between the DNA molecule and the recombinase. Rejoining of the DNA is performed via the reverse reaction. Consequently, no ATP is needed to perform this reaction. Despite the apparent similarity of the two recombination mechanisms, the serine and tyrosine recombinase families evolved separately; their mechanisms are different. Tyrosine recombinases form Holliday junctions and break one DNA strand at a time, while serine recombinases form simultaneous double-stranded breaks (Figure 1.12).

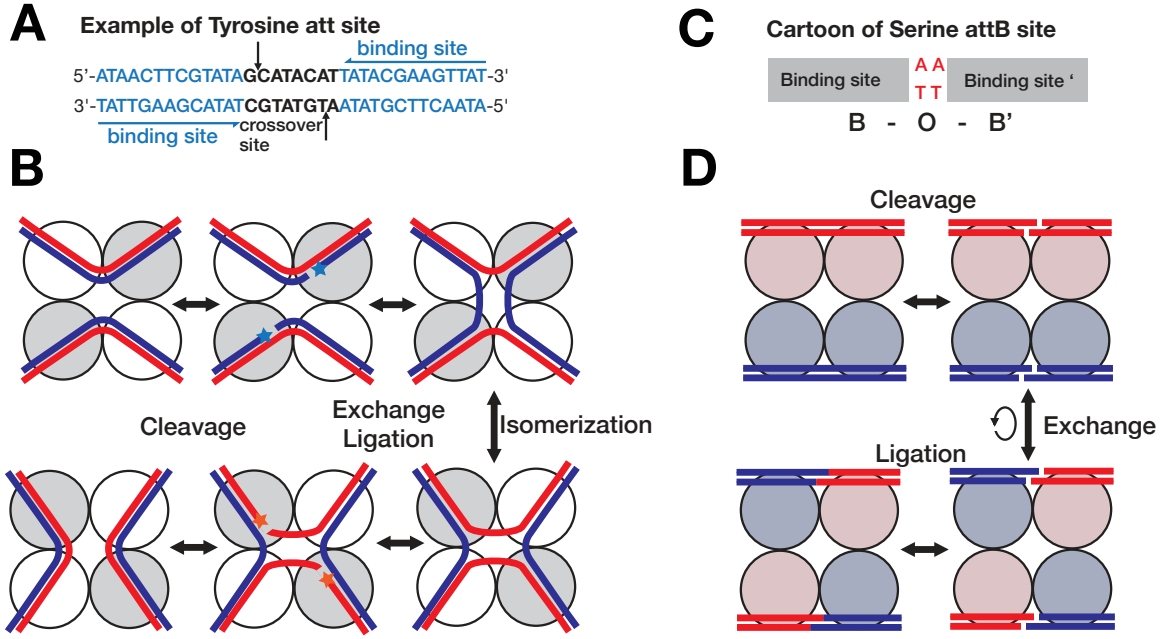


Figure 1.12: Mechanism of tyrosine and serine recombinases. A - A part of the LoxP-Cre site with the Cre binding site (in blue) and the crossover site (in black). The arrow corresponds to the cutting sites. B - Cartoon of tyrosine recombination mechanism. The blue and red lines are two DNA strands. The active recombinases of the 4 tetrameric complexes are represented in grey. In the phase of isomerization, this activity switches. C - Cartoon of the serine recombination site composed of two partially complementary binding sites and a two bp core site. D - Cartoon of the serine recombinase mechanism. The blue and red colors are used to illustrate the exchange reaction.

1.3.1.2 Tyrosine recombinases

For tyrosine recombinases, the minimal recombination sites are comprised of a pair of inverted enzyme binding sites separated by 6 to 8-base pair spacers. Then, the cleavage of DNA is performed at the 5' end of the spacer. The synaptic complex is formed as a tetramer. At each site, a dimer is formed and the 6 bp crossover site must bend to permit formation of the dimer. According to the bend of the crossover site, only one DNA strand is available for cleavage and the protomer in the 5' end of this strand is active. Then, within the synaptic tetramer, alternating protomers are active one at a time. In each duplex, one DNA strand is cleaved

by the nucleophilic tyrosine creating a 3' phosphotyrosine linkage between the DNA and the protein and a free hydroxyl group at the 5' end of the DNA. Then, the free 5' end of the DNA molecule attacks the 3' phosphotyrosine of the other DNA molecule forming a Holliday junction. For the second step, the inactive protomers become active, and the process is repeated with the second DNA strand.

The crossover site can adopt two different bends to permit the contact between the two protomers binding to the same site. This conformation determines which strand will be cleaved first. The recombination is efficient only when both sites adopt the same bends leading to antiparallel orientation of the crossover site in the synaptic complex. Indeed, with parallel orientation, the ligation will not occur as a mismatch is formed, which can lead to the reverse reaction. In the simplest cases, the two recombinase sites in the reaction are identical (such as for Flp and Cre). Then, the reverse reaction is as favored as the forward reaction; consequently, the reaction reaches equilibrium when there is 50% of substrate and product. To have unidirectional reactions, accessory factors are used.

The simple Flp- and Cre-recombination reactions are not unidirectional. Moreover, Flp does not have any preference for the location of the first DNA cleavage - either end of the spacer. The antiparallel reaction is then not favored. Cre-recombination preferentially cleaves first at the GpC end of the site spacer. This catalytic preference is due to one DNA bend which is favored, likely because of constraints on the DNA sequence flexibilty or on the protein-DNA interaction [Guo 1997].

The homology between the two recombinase sites is essential for ligation of the two DNA strands after DNA break and strand exchange. For Cre recombinase, it has been shown that the identity of the 6 central base pair is essential. Mutations of this crossover site were performed [Lee 1998] and showed that mutated sites had a decreased recombination efficiency and were approximately orthologues. The two sites, 2272 and 5171, are the most used mutated sites due to their good orthogonality and efficiencies.

Using accessory factors, the recombination reaction can be unidirectional, such as for lambda Int [Nash 1981]. Lambda Int catalyzes the integration of the lambda phage genome into the *E. coli* genome. In this system, the Integration and excision of the phage genome must be well regulated. To regulate the directionality of the recombination, lambda Int contains an additional domain at its N terminus. The domain binds to additional DNA sites placed on both sides of the crossover site of the attP site. The additional protein domain and additional DNA binding sites (arm sites) specify the bend of the attP sites for integration. The attB site does not possess any additional binding site, but specifically binds in one orientation due to its DNA sequence. The integration leads to attL and attR sites composed of half attP and half attB sites, such as B-0-P' for attL and P-0-B' for attR. Consequently, as no integrase sites are composed of the two arm sites, no reverse reaction can occur. Excision of the DNA is mediated via Xis enzyme, which changes the synaptic complex conformation and recognizes the attL-attR sites

instead of attB-attP sites. The N-terminus domain of lambda Int, the non-symmetricity of the two integrase sites, and the use of an excision cofactor allow directionality of the recombination mechanism in tyrosine recombinases.

1.3.1.3 Serine recombinases

Serine recombinases are mainly composed of two domains, a binding domain and a catalytic domain, differing significantly between recombinases. For the $\gamma\delta$ resolvase, the catalytic domain positioned in the N-terminal is linked by an alpha-helix (E-helix) and an unstructured segment to a helix-turn-helix DNA-binding domain at the C terminus [Yang 1995]. The H-T-H domain is responsible for the site recognition. In the Tn3 recombinase, it has been replaced by a Zinc finger recognition [Akopian 2003]. For the different serine recombinases, the H-T-H DNA binding domain can be positioned either in the C-terminus or N-terminus of the catalytic domain. In all recombinases, the catalytic domain with its nucleophile serine is conserved in addition to an E-helix following the binding domain. According to the resolved structure [Li 2005], the catalytic domain and E-helix are positioned in the synapse of the tetramer domain and the DNA-binding domain with the DNA positioned outside of the synaptic domain. The synaptic complex is formed with two recombinase subunits per site forming a tetramer. The four subunits are activated at the same time and break the two strands of the two sites, performing two double-strand breaks. Each recombinase subunit is then bound by a phosphoserine linkage to the 5' end of the broken DNA strand where all 3' hydroxyl ends are free ([Reed 1984]. The exchange of strands is performed by a relative 180°rotation of half of the complex. The rotation is usually right handed to relax the negative superhelicity of the DNA; however, depending on the DNA conformation, the rotation can occur in both orientations. Then, the 3' hydroxyl group attacks the 5' phosphoserine to permit re-ligation of the DNA.

The two DNA strand breaks leave 2-bp 3' single strands during the strand exchange [Reed 1981]. These 2-bp have to be conserved between the two sites to allow re-ligation by complementarity of the two strands. As for tyrosine recombinase with its 6-8 bp crossover site, modification of the 2-bp serine-recombinase crossover site permits engineering of orthogonal recombination sites. However, a mismatch of the 2-bp is only recognized after cleavage and strand exchange, when the rejoining is attempted. In the presence of a mismatch, the recombinase has to perform a new round of strand exchange to return to the initial configuration.

Based on this simple mechanism, excision vs. inversion can only be identified from asymmetric 2-bp central sequences and intramolecular vs. intermolecular exchange cannot be identified. However, to perform the desired recombination reaction, some regulations appear at the formation of the synaptic complex to promote its formation in a specific conformation.

The resolvase family specifically performs excision using recombination sites composed of multiple binding sites [Mouw 2008] [Yang 1995]. The $\gamma\delta$ recombinase site is composed of three

double-binding sites, head-to-head, separated by different size spacers. These multiple binding sites force one specific synaptic conformation, and therefore only the excision reaction is mediated.

The Hin and Grin invertases mediate naturally the inversion of promoters or coding sequences to switch gene expression. The specification of their reaction is performed using an additional protein: factor for inversion stimulation (Fis). A DNA sequence specific for Fis binding called an enhancer is needed for recombination. This enhancer is composed of two binding sites for the Fis dimer separated by 48 bp. The complex called invertasome is composed of the four recombinase subunits, the Fis dimer, the enhancer, and the two recombinase sites. The substrate DNA must be supercoiled in order to permit Fis-Hin interaction [Heichman 1990]. The formation of the invertasome forces the inversion reaction to occur and moreover, Hin recombination is inactive without the Fis dimer and the enhancer in *cis* configuration.

Finally, our subfamily of interest is the serine integrases. Serine integrases mediate, as with lambda integrase, the integration and excision of a phage genome into a bacterial genome. Therefore, it has to distinguish intermolecular recombination for integration from intramolecular recombination for excision. Lambda integrase succeeds to differentiate and regulate these two reactions using accessory sites in attP and cofactors. Serine integrase uses a totally different mechanism to distinguish the two reactions. The best studied integrases are ϕC31 from the *Streptomyces* phage and Bxb1 and ϕRv1 from mycobacteriophages. For serine integrases, attB and attP sites are around 40 bp. In contrast to other recombinases, attB and attP sites have highly variable sequences, except for the conserved 2-bp crossover site. Serine integrase alone mediates the integration of the phage genome in bacterial genome via attP-attB recombination. Indeed, integrase stably binds to attL and attR sites, but it seems that only attP and attB sites permit the formation of an active complex. To perform excision of phage DNA from the bacterial genome, an excision cofactor, called Xis or RDF (for recombination directionality factor), is needed. The RDF with the integrase allows catalysis of the attL-attR complex and inhibits the catalysis of the attB-attP complex. The RDF does not need any extra DNA sequences and interacts with the integrase dimers to permit formation of an active synapse with attL-attR. When the RDF is present, the stability of the synapse switches from the attB-attP complex to the attL-attR complex.

1.3.1.4 Highlight on the serine integrase mechanisms and specificities

Serine integrases are a subfamily of serine recombinases, also called Large Serine Recombinases. They have an N-terminal catalytic domain similar to other serine recombinases, but a much larger C-terminal domain (of 300 amino acids for A118). This different C-terminal domain seems to be responsible for the binding to the attachment sites and for the catalytic specificity of serine integrases. Serine integrases unidirectionally catalyze attBXattP recombination without the need of additional binding sites or co-factors. And in the presence of RDF, the catalytic activity

of the attLXattR site is actived while attBXattP is inactived. According to recent partial crystal structures, it seems that this catalytic specificity is due to the structural conformation of the C-terminal domain, which is highly dependent upon the binding to the different sites and influences the formation of the synaptic domain [Van Duyne 2013].

Only partial crystal structures are available for serine integrases, such as the catalytic domain of the bacteriophage TP901-1 integrase [Yuan 2008] and the C-terminal region of bacteriophage A118-like integrase bound to an attP half-site [Rutherford 2013].

The most characterized serine integrases are Bxb1, Tp901-1, PhiC31, and A118. Using bioinformatic tools, Yang and colleagues identified 4,000 putative serine integrases [Yang 2014]. Eleven orthogonal integrases were characterized with their attachment sites. However, a reduced number of RDF have actually been identified and no sequence conservation has been found between identified RDFs (such as for Bxb1 [Ghosh 2006] and PhiC31 [Khaleel 2011]) (List of integrase and RDFs: Table 1.2). Stoichiometrically, one RDF is needed per integrase as different stoichiometries lead to different switch probabilities [Bonnet 2012]. Fusions of RDFs to integrases were engineered to obtain reliable unidirectional attLxattR recombination [Olorunniji 2017]. As serine integrases do not require cofactors, they are more easily transferable to other organisms, such as mammalian cells [Keravala 2006].

Integrase	RDF	Reference for RDF
Bxb1	gp47	[Ghosh 2006]
Tp901-1	Orf7	[Breüner 1999]
PhiC31	gp3 interchangeable with PhiBT1	[Khaleel 2011]
A118	Gp44	[Mandali 2017]
PhiRv1	xis	[Bibb 2005]
PhiBT1	gp3 interchangeable with PhiC31	[Zhang 2013]
Int2-5, 7-13	No yet identified	

Table 1.2: Serine-integrases and their identified RDFs with corresponding references.

Orthogonal sites for the same integrase can be engineered by changing the two central base pairs where the double strand break occurs. Re-ligation of recombined sites is only possible between two sites with identical central base pair couplets. With a divergent central nucleotide sequence, the double-strand break can occur but the sites will re-ligate in their original conformation [Ghosh 2008].

As the palindromic 2-bp central sequences lead to no specificity between excision and inversion, potentially six orthogonal pairs of sites can be constructed for each integrase using the following 2-bp sequences: TT/AA, CT/GA, GT/CA, TG/AC, CC/GG and TC/AG. Such orthogonality was applied to attB/attP sites and also to attL/attR sites using PhiC31 integrase [Colloms 2014]. The use of six integrase-site pair variants for DNA assembly leads to only 18%

of correctly assembled pathways [Colloms 2014] under selective pressure. Consequently, the integrase-site pair variants seem to not be fully orthogonal.

1.3.2 Recombinases as a tool for DNA editing

Using their natural function to integrate, excise, and invert DNA or even resolve DNA concatenation, recombinases have been used for 30 years as a tool for *in vitro* cloning, genome modification, and cell-lineage construction [Sauer 1994]. At first, tyrosine integrase were principally used, as more was known about them than on serine recombinases. Many systems were based on Cre and FLP recombinases, which do not required cofactors and can functioned in various organisms, such as mammalian cells. Then, as the mechanism of serine integrase became better understood and a larger set was characterized, increasing number of systems were developed based on serine integrases. They have proved to be great tools as their recombination is precise, specific, and unidirectional, and they work very efficiently in a wide range of organisms.

Recombinases can be used *in vitro* as a restriction/ligation mechanism, and they are more efficient than usual restriction and ligation enzymes but nevertheless require larger recognition sites. First, cloning vectors were designed using Cre recombinase [Sauer 1988]. Later, the GATEWAY system was developed based on lambda integrase with its HIF cofactor (Figure 1.13A). Up to 6 orthogonal pairs of att sites were used to permit single-step gene assembly of up to 5 fragments [Hartley 2000] [Cheo 2004]. In comparison to other strategies such as Gibson assembly, the cloning leads to fewer mutations but requires large overlap sequences of around 200 bp (att sites) between fragments. Using serine integrases and integrase site variants, other *in vitro* cloning systems similar to GATEWAY were developed, based either on PhiBT1 integrase, site-specific recombination tandem assembly (SSTRA) [Zhang 2011], or on PhiC31 integrase, serine integrase recombinational assembly (SIRA) using integrase-site variants [Colloms 2014]. A major benefit of serine integrases is the requirement of only small integrase sites (40bp), which can simply be added by PCR. However, the use of serine integrase site variants with modification of 2 bp crossover sequence have shown only an 18% correct sequence for 5 part assembly, likely due to imprefect orthogonality of integrase site variants.

Additonally, recombinases have been used for genome engineering. First, recombinases were used to remove selective markers after integration of DNA via homologous recombination. For example, in mammalian cells selective markers were surrounded by FLP integrase sites which mediated their excision by expression of FLP recombinase [Fiering 1993]. Landing pads for site-specific integration were also developed (Figure1.13B). To do so, cell-lineages were first engineered via homologous recombination to place an integrase site at a specific location. Then, any cassette surrounded with the complementary integrase site can be integrated at this specific locus. These landing pads were used in *Drosophila* to study positional effect on transgene expression. A collection of strains with landing pads at different loci with integration mediated through PhiC31 mRNA was developed by Bischof and colleagues [Bischof 2007]. Similarly, using

R4 integrase, Invitrogen developed a Jump-in Targeted Integration System with pre-integration R4 attP sites at known genomic loci of mammalian cells [Scientific 2017]. Moreover, pseudo PhiC31 att sites were found in the human genome [Chalberg 2006], and while recombination is less efficient with these pseudo sites, it permits integration of transgenes without previous cell modification. Using this approach, pseudo PhiC31 att sites were applied to gene therapy techniques [Olivares 2002] [Ortiz-Urda 2002].

Similarly to landing pad, recombinase-mediated cassette exchange (RMCE) is also used to target genome integration (Figure 1.13C). For RMCE, the cassette for integration is flanked through two integrase sites and the complementary sites are positioned in the target genome locus. This technique was first developed using FLP via two mutant sites [Schlake 1994]. Then, it was adapted to serine integrases, which are better candidates than FLP as recombination is highly directional and att sites are relatively small. A variation of RMCE, dual integrase cassette exchange (DICE), uses a pair of orthogonal serine integrases [Zhu 2014].

Recombinases were also used to induce gene expression at specific time. A first synthetic system for *E. coli* used lambda integrase to invert a promoter and induced gene expression [Podhajska 1985] (Figure1.13D). By changing the site orientation, DNA inversion through lambda integrase was engineered. In this plasmid, lambda integrase expression was mediated through a heat pulse induction, which triggered promoter inversion and expression of the output gene placed downstream. After the pulse of integrase expression, the output gene is constitutively expressed. It therefore permits induction of gene expression without the constant use of chemical inducer and constant expression of regulators. Similarly, Cre recombinase was used in mice to permit activation of dormant transgene by excision of STOP sequences using Cre recombination. This strategy is named recombination activation of gene expression (RAGE) [Lakso 1992, Pichel 1993] (Figure 1.13E). Excision of STOP sequences was performed via crossing the dormant transgenic mouse line with Cre-expressing transgenic lines, permitting activation of a large-tumor antigen.

Similarly, FLP integrase expressed through a heat-pulse induction was used to randomly induce the expression of a specific gene. By variation of the intensity and length of heat-pulse, a proportion of cells express the specific gene, generating *Drosophila* mosaics [Struhl 1993]. Other systems used recombination to generate random genetic events. Recently Cre recombination was used in the design of a synthetic yeast genome to enable rapid evolution of genome in a random manner by placing several Cre recombination sites, known as synthetic chromosome rearrangement and modification by loxP-mediated evolution, or SRaMbLE [Shen 2016]. Additionally, Livet and colleagues developed a system to identify individual neurons using random genetic events. In this system, various fluorescent proteins are placed between Cre and FLP attachment sites. By expressing the recombinases, random recombination events occur, leading to various patterns of fluorescent protein expressions creating a specturm of various colors. Up to 10 colors were obtained using this BRAINBOW system [Livet 2007].

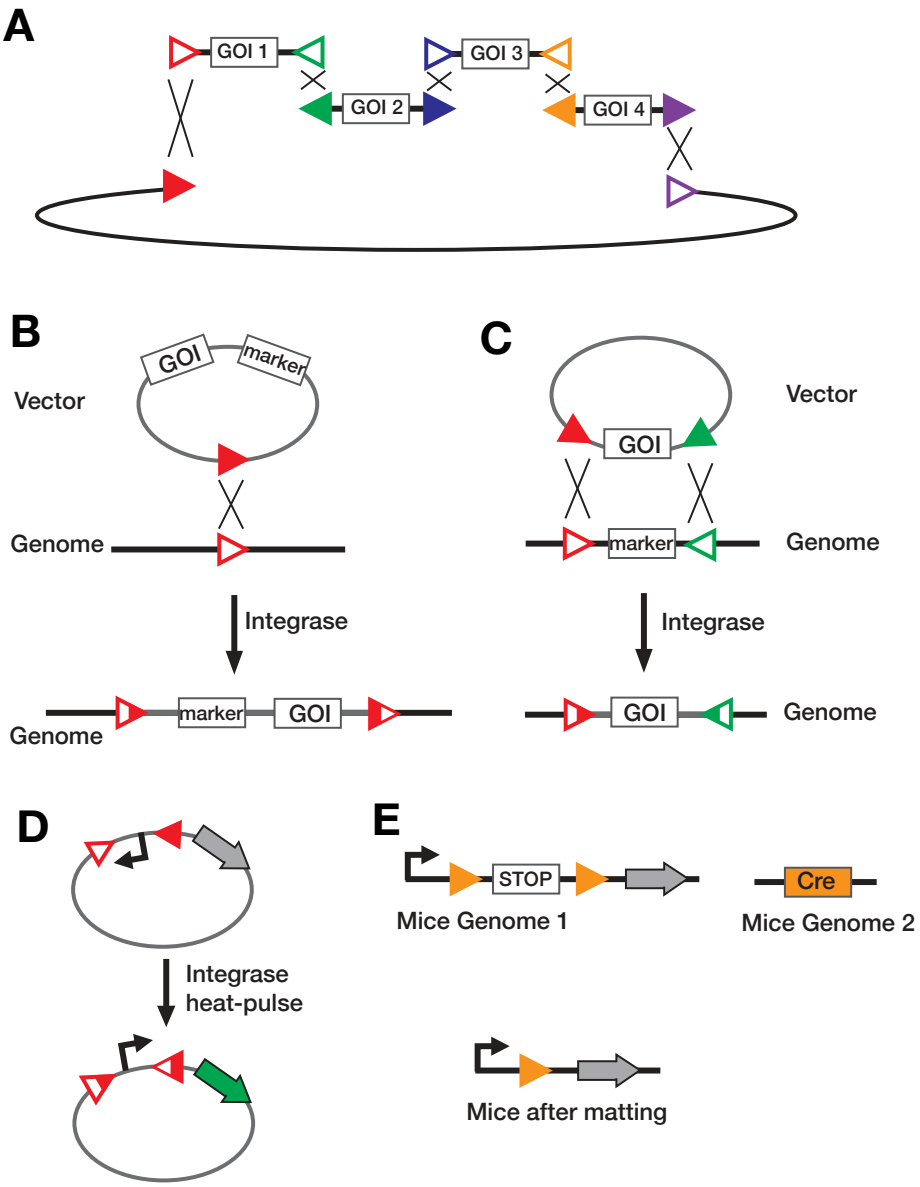


Figure 1.13: **Application of recombinases for DNA editing.** A - *In vitro* assembly of multiple DNA fragments via recombinases: GATEWAY system. B - Landing pad for site-specific integration. C - Recombination-mediated cassette exchange. D - Gene-expression induction via promoter inversion mediated by integrase induced by an heat pulse. E - Recombination Activation of Gene expression (RAGE) in mice.

1.3.3 Recombinases as a tool for logic implementation

As explained previously, recombinases permit implementation of genetic switches to turn on gene expression or to generate random gene-expression patterns. Based on this simple system, more complex circuits have been implemented. Recombinases were used to solve computation problems such as the burnt pancake problem and Hamiltonian path problem, and to implement logic circuits, such as sequential and combinational logic circuits. Here, we detailed how the

mechanism of recombinases can be hijacked to implement complex computation programs.

In most systems, the output of the circuit is the expression of an output gene. Therefore, promoters, terminators, and genes are placed between integrase sites to control the output gene expression by recombination events, wich can be either stochastic or deterministic.

Stochastic recombination events were used to generate all possible combination of various genetic parts. To do so, several integrase site copies are placed in the same sequence, then various recombination patterns can occur between the different integrase sites, leading to different DNA sequences. This strategy was used to solve the burnt pancake problem (BPP) [Haynes 2008] and a 3-node Hamiltonian path problem via the Hin tyrosine recombinase [Baumgardner 2009]. In these two papers, a low rate of inversion was mediated through the use of multiple hixC sites, then each recombination occured at a certain probability, leading to the generation of a set of various sequences. Promoters, genes or split-genes were positioned between hixC sites such that recombination led to various gene expression profiles. A similar strategy was used in BRAINBOW to generate stochastic patterns of fluorescent protein expression [Livet 2007] using Cre recombinase (Figure 1.14). Different Brainbow designs were implemented, including one design where orthogonal Cre sites in excision orientation overlap with second pair of sites such that the recombination event occuring first excises a site of the second pair. Depending on the stochasticity of recombination, various gene-expression patterns are generated. Another design uses 4 loxP sites positioned to obtain four expression pattern possibilities depending on inversion and excision recombination events.

By placing integrase sites around gene-expression regulatory elements, such as promoters, output gene expression can be irreversibly switched ON through integrase expression. The system from Podhajska et al. [Podhajska 1985] corresponds to the implementation of a one input Boolean function, considering the input ON when the input has been present. By placing these genetic switches in series, in parallel or in layers, more complex Boolean function can be implemented.

Following this principle, 2-input Boolean logic functions were implemented in bacteria using serine integrases. Bonnet and colleagues, and Siuti and colleagues, implemented all 2-input logic functions in single cells using either Bxb1/Tp901-1 integrases or Bxb1/PhiC31 integrases [Bonnet 2013, Siuti 2013]. They used serine integrases due to their high specificities, the unidirectionality of the recombination events, the possibility to perform inversion or excision depending on site orientations, and the possibility to transfer system to various organisms. In these system, the presence of an input induces the expression of one serine integrase and therefore the recombination between one pair of integrase sites. Bonnet and colleagues circuits are based on an asymmetric terminator surrounded by integrase sites. By nesting integrase sites, changing orientation of sites, and placing up to two terminator switches in series, they implemented all the 2-input logic gates in *E. coli* (Figure 1.15A). Siuti and colleagues placed between integrase sites either promoters, terminators, or genes. Using only integrases, which mediate irreversible

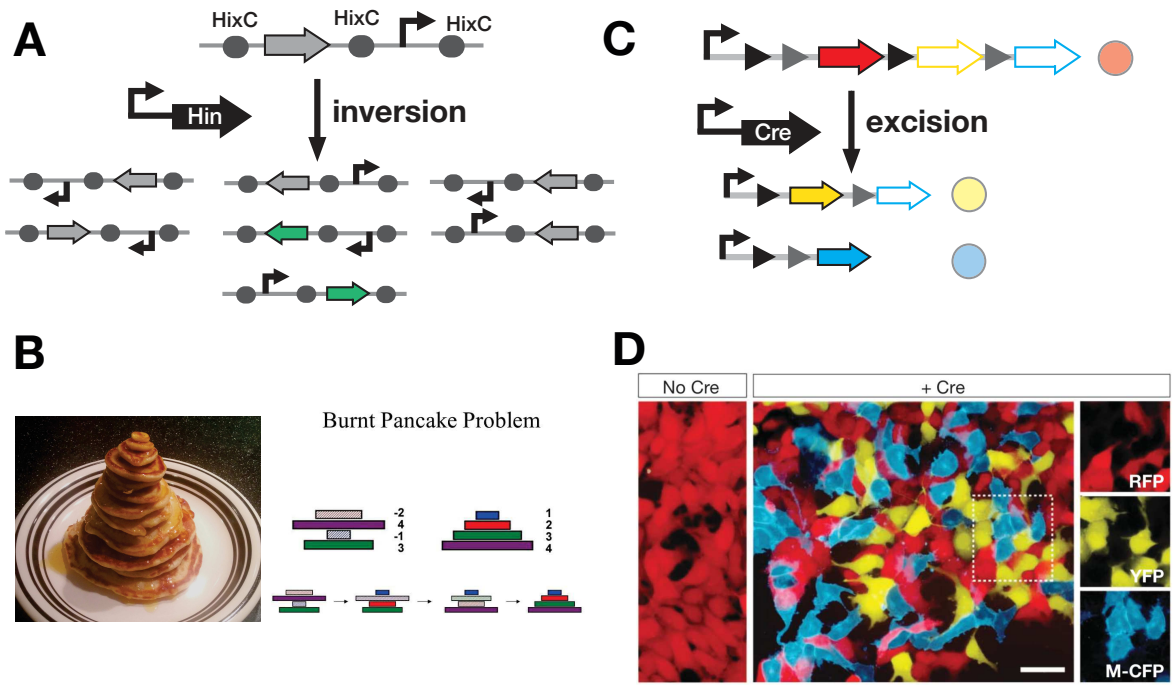


Figure 1.14: **Use of stochastic recombination events to generate various genetic circuits.** (A) (B) Burnt-pancake problem using Hin recombinase. Based on the low inversion efficiency of HixC site of Hin recombinase, all the possible combinations of the two parts are obtained. The output gene is expressed only when the promoter and gene are in the correct orientation, corresponding to the solution for the 2-part BPP. [Haynes 2008] (C) (D) Cre recombinase and multiple LoxP sites are used to generate stochastic pattern of fluorescent protein expressions to label cells with various color. (D) corresponds to a figure from [Livet 2007].

genetic switches, these circuits implement asynchronous one-shot Boolean logic; the input is considered ON if it have been present and system cannot be reset.

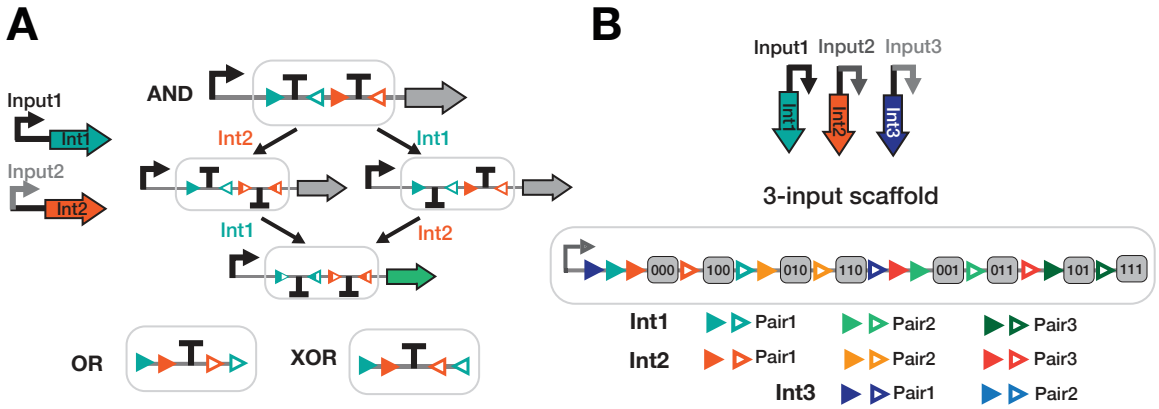


Figure 1.15: **Recombinase-based logic circuit designs.** (A) Bonnet et al. design using composition of terminator-based elements. (B) Weinberg et al. design using excision, integrase-site variants and one locus for gene expression per input state.

Similarly, Weinberg and colleagues implemented asynchronous one-shot Boolean logic in mammalian cells [Weinberg 2017]. The BLADE system (for Boolean Logic and Arithmetic through DNA Excision) uses serine integrases to mediate inversion and tyrosine recombinases to mediate excision. Bxb1 and PhiC31 integrases are used for inversion and Cre, Flp, and VCre recombinases are used for excision. For tyrosine recombinases, three orthogonal mutant siteswere characterized for each recombinase: lox sites (loxP, lox2272, loxN),FRT (FRT, F3, F14) and Vlox (VloxP, Vlox2272). These sets of mutant sites permit various recombination events responding to the same recombinase and therefore to the same input. Then, based on this set of integrase site pairs, Weinberg and colleagues constructed a 3-input scaffold based on the expression of a different GOI position at each Boolean state (Figure 1.15B). This 3-input scaffold permits implementation of all 3-input Boolean functions by placing a gene at the GOI locus corresponding to each ON state. It was at this time the only systematic framework for design of integrase-based logic circuits in living organisms. The design workflow is simple but leads to large constructions, and for each ON state a gene is required and for each OFF state a terminator is needed. Scaling up this design to higher numbers of inputs seems challenging as the size of the circuits increase exponentially with the number of inputs, as the number of required orthogonal integrase sites per inputs. However, BLADE shows the capacity of integrase-based logic circuits to permit implementation of logic within compact genetic circuits, specifically within a single construction.

Another possibility for implementation of Boolean logic circuit is to layer serine integrases [Yang 2014], similarly to repressor-based logic circuits. Layering serine integrase remains challenging as switches are highly sensitive to integrase gene-expression leakage. Moreover, by layering recombinases, the design loses the ability to have a compact and single-layer implementation, which is one of the primary benefits of a recombinase based design. Additionally, in Yang et al., the integrase switch was placed as the last layer of repressor-based logic circuits, which permits implementation of memory in repressor-based logic circuits that lack it.

Most of sequential logic implemented in living organisms is based on recombinases as it permits a direct and simple recording of input occurence due to the irreversibility of some recombinase switches. Focusing here on recombinases, I review the *in vivo* implementation of sequential logic systems in Annex B.

By overlapping integrase site pairs corresponding to different inputs, the order of occurrence of events can be differentiated as recombination events are interdependent. The design principle is similar to the BRAINBOW design where integrase site pairs are overlapping; however, for sequential logic implementation, the system behavior is deterministic as integrase sites respond to different inputs. First, Ham and colleagues constructed an history-dependent system using FimB and Hin integrases by overlapping integrase sites [Ham 2008a]. Inputs correspond to the expression of FimB and Hin integrases and both integrases mediate DNA inversion. As the two sites are overlapping, different DNA sequences are obtained depending on the order

of occurrence of inputs. However, as FimB inversion is not unidirectional, there is a 50/50 proportion of FimB switch. Despite its interesting design, the system is not fully functional, probably due to the use of FimB.

Hsiao and colleagues constructed a similar target using serine integrases [Hsiao 2016] (Figure 1.16A). In this design, inputs control expression of Bxb1 and Tp901.1. In the target, a Bxb1 integrase site pair is in excision orientation and a Tp901.1 integrase site pair in inversion orientation. As pair of sites are overlapping, expression of Bxb1 integrase first leads to excision of one Tp901.1 site. Then, if Tp901.1 integrase is expressed after Bxb1, no switch occurs. However, if Tp901.1 is express first, it causes inversion of one Bxb1 site, such that expression of Bxb1 after Tp901.1 leads to inversion of DNA sequences. Using this temporal target, 4 different history-dependent states can be differentiated on the DNA and 3 states via gene-expression by placing one promoter and one terminator between the sites and one gene at each extremity of the temporal target. If the two inputs occur within a short delay, a mixed population will be obtained as not all recombinations will have the time to occur. Therefore, the quantification of the proportion of each DNA state permits to determine the time between the occurrence of each event.

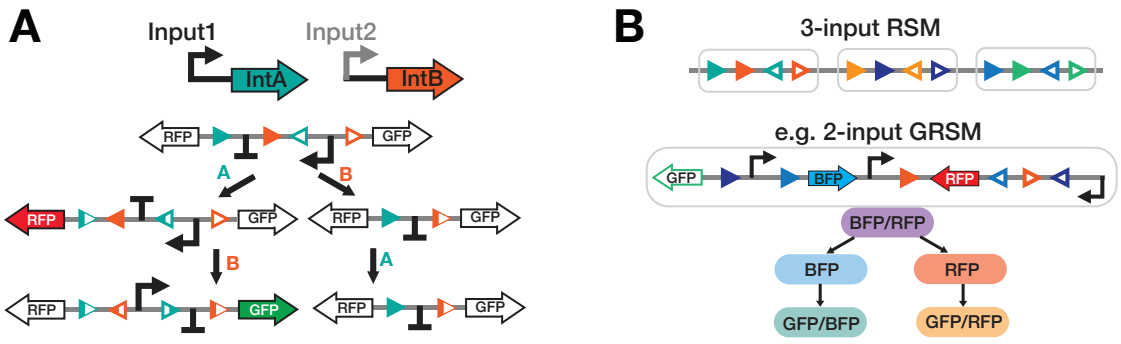


Figure 1.16: **Serine-integrase history-dependent circuits.** (A) 2-input temporal target from Hsia et al. 2016. (B) 3-input recombinase-based state machines with DNA as output and 2-input with gene expression as output.

Scaling up this sequential circuit design, Roquet and colleagues implemented 2- and 3-input history-dependent circuits, termed RSM for recombinase-based state machines (Figure 1.16B). Their design is based on the same 2-input temporal target as Hsiao and colleagues. For 3-input, three targets responding to each 2-input combination are placed in series (Figure 1.16B upper panel). Bxb1, Tp901.1 and A118 serine integrases respond to each input and for each integrase, two integrase site pair variants are used to have independent 2-input targets. This design permits to record the history of occurrence of 3 events, as each 16 sequential input state leads to a different DNA sequence. The state of the system can then be read via DNA sequencing. By placing promoters, terminators, and genes between integrase sites, history-dependent gene-expression programs can be implemented (Figure 1.16B, lower panel). Based on their 3-input

target composed of only integrase sites, Roquet and colleagues generated all possible 3-input gene-expression RSMs by placing gene-expression parts between integrase sites. Via brute-force generation, they created a database of gene-expression programs with the various GRSMs. Based on their paper, it is not clear if all history-dependent gene-expression programs can be implemented via this design strategy. And as this design strategy is based on brute-force, its seems limited to generate all parts combination for a high number of inputs. It is unfortunate that a web interface has not been created to permit to the many systematic designs of history-dependent gene-expression circuits, as their design is not straightforward.

On the experimental side, they characterized a set of 2- and 3-input multi-output history-dependent logic circuits. This design is theoretically scalable to 7 inputs as the maximum number of orthogonal site variants is 6. However, it seems unlikely that this design is feasable for a high number of inputs, as for 7 inputs it will lead to a DNA sequence with 7 times 6 repetitions of 40 base pairs. Additionnaly, as shown by Colloms and colleagues [Colloms 2014], 2bp integrase site variants are not fully orthogonal such that the use of so many variant sites will probably lead to non-specific recombination events.

All serine integrase-based genetic circuits are one-shot as serine-integrase switch is uni-directional. However, using recombinase directional factor (RDF), a rewritable switch was implemented using Bxb1 and Bxb1-RDF, called the rewritable digital data storage (RAD) [Bonnet 2012]. The RAD target is composed of a promoter surrounded by Bxb1 integrase sites in inversion orientation between a GFP and a RFP gene. Bxb1 integrase expression permits switching from RFP to GFP expression and Bxb1-RDF coupled with Bxb1-integrase permit resetting to RFP expression. However, reset necessitates tight control of integrase and RDF stoichiometry, as integrase alone mediates attB-attP recombination and when coupled with RDF mediates attL-attR recombinase. Theoretically, the combination of RAD circuits with a repressor and activator system permits implementation of an activation-repression toggle flip-flop circuit, corresponding to the output turned ON in presence of one input activation and OFF when input have been off and it is on again [Subsoontorn 2012a]. This toggle flip-flop circuit can be used to implement a combinatorial asynchronous counter. In general, a flip-flop circuit is the basic component needed for implementation of complex synchronous sequential circuits.

Friedland and colleagues used the tyrosine recombinases Flp and Cre to implement a synthetic circuit which “counts”, i.e. output is expressed after three induction pulses [Friedland 2009]. The objective of this circuit is to count the number of pulses of a chemical molecule, arabinose. The first arabinose pulse induces the expression of Flp recombinase, which mediate inversion of its gene coding sequence and an arabinose-inducible promoter that promotes expression of Cre recombinase. This schematic is repeated for the second arabinose pulse with the Cre recombinase. At the third arabinose pulse, the GFP output gene is expressed, which permits to count until three. However, the system is unable to distinguish several pulses

from a long induction. Therefore, continuous arabinose induction will turn ON GFP expression. This system is consequently a counter of input-pulse and not strictly speaking a counter.

Author	**Logic**	**Recombinases**	**input - integrase**	**type of design**	**Max number of inputs**
Bonnet	Asynchronous one-shot Boolean	Bxb1-Tp901.1 serine integrases	One integrase and one pair of site per input	Single-cell	2 inputs
Siuti	Asynchronous one-shot Boolean	Bxb1-PhiC31 serine integrases	One integrase and one pair of site per input	Single-cell	2 inputs
Weinberg	Asynchronous one-shot Boolean	Bxb1-PhiC31 serine integrases, Cre, Flp, Vcre tyrosine recombinases	One integrase and up to 3 pair of site per input	Single-cell	3 inputs
Roquet	Asynchronous one-shot sequential	Bxb1, Tp901.1, A118 serine integrases	One integrase and up to 6 pair of site per input	Single-cell	3 inputs

Table 1.3: **List of recombinase-based logic circuits of interest**, with the corresponding papers, the type of logic implemented, the recombinases used, the number of integrase or integrase-site variants per input, the type of implementation (single or multi-cell) and the maximum number of inputs computed.

A large set of recombinase-based circuits have been developed in the last decade (Table 1.3). Recombinases permit engineering of a large range of computation circuits. Most systematic work has been performed on the implementation of asynchronous one-shot Boolean and sequential logic, however, circuits are still limited to processing a maximum of 3 inputs [Roquet 2016] [Weinberg 2017] and remain proof-of-concept.

1.4 Thesis objectives

The objective of my thesis is to increase the computation power of recombinase-based logic circuits. I aimed at developing an automated design and systematic design framework enabling researchers to simply implement logic circuit into a large range of organisms and of inputs.

Below are the circuit specification that I aimed at developing:
- Compact: reduce the number of parts needed.
- Automatic design: theoretical design performed via a web-interface.
- Transferable: implementable in various organisms.
- Scalable: for an increasing number of inputs.
- Complete: for all logic functions.
- Accessible: non-experts can use these tools and apply them to other applications.
- Reliable: behave as expected.
- Reusable: parts developed can be used for construction of other circuits.

During my thesis, I developed serine recombinase-based logic circuits for asynchronous single-shot Boolean logic and history-dependent logic.

I implemented asynchronous Boolean logic operating in a multicellular system (Chapter 2). I developed an automated design framework for Boolean function in multicellular systems and engineered a set of 14 logic devices that allow the implementation of all 4-input Boolean functions.

Similarly, I developed multicellular history-dependent logic programs (Chapter 3). I automated the design of these circuits and engineered a proof-of-concept of the implementation of up to 3-input history-dependent gene expression programs in *E. coli*.

Finally, I developed minimization schemes for single-cell recombinase logic circuits (Chapter 4). This design is complementary to the multicellular design of Boolean logic in Chapter 2, as single cell designs emphasize compactness over reusability and easy implementation. We generated a database of all possible logic circuit designs in single-cell systems using a combinatorial approach and developed a web-interface called Recombinator. Finally, I developed a strategy to experimentally characterize a reduced set of construction to determine the feasibility and completeness of the implementation in living organisms.

Chapter 2

Boolean logic in multicellular consortia using recombinases

Contents

In this chapter, I will present my work on the implementation of Boolean logic in multicellular consortia using recombinases. This work is divide in two parts: (1) the development of an automated design framework and (2) the experimental implementation of this design.

Many people contributed to this work. Jerome Bonnet and myself were at the origin of the project. Michel Leclere and Federico Uliana participated in fruitful discussions on the circuit designs. I came up with the design workflow and its automatization and created the Python software. Violaine Moreau and Laurent Bonnet were involved in the creation of the CALIN website, and, I designed the biological construction. The characterization of biological parts and implementation of Boolean logic circuits were performed by Pauline Mayonove, myself, and Chloé Thailhades.

2.1 An automated design framework for multicellular recombinase logic

We published the automated design framework for multicellular recombinase logic in ACS Synthetic Biology. Following the full paper and the supplementary data can be found in Annex D.

Abstract: Tools to systematically reprogram cellular behavior are crucial to address pressing challenges in manufacturing, environment, or healthcare. Recombinases can very efficiently encode Boolean and history-dependent logic in many species, yet current designs are performed on a case-by-case basis, limiting their scalability and requiring time-consuming optimization. Here we present an automated workflow for designing recombinase logic devices executing Boolean functions. Our theoretical framework uses a reduced library of computational devices distributed into different cellular subpopulations, which are then composed in various manners to implement all desired logic functions at the multicellular level. Our design platform called CALIN (Composable Asynchronous Logic using Integrase Networks) is broadly accessible via a web server, taking truth tables as inputs and providing corresponding DNA designs and sequences as outputs (available at `http://synbio.cbs.cnrs.fr/calin`). We anticipate that this automated design workflow will streamline the implementation of Boolean functions in many organisms and for various applications.

ACS SyntheticBiology
Research Article
Cite This: ACS Synth. Biol. 2018, 7, 1406−1412
pubs.acs.org/synthbio

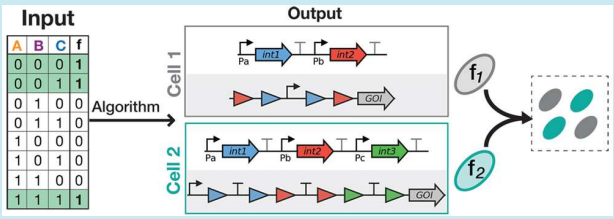
An Automated Design Framework for Multicellular Recombinase Logic

Sarah Guiziou,[†] Federico Ulliana,[‡] Violaine Moreau,[†] Michel Leclere,[‡] and Jerome Bonnet*[,†]

[†]Centre de Biochimie Structurale (CBS), INSERM U1054, CNRS UMR5048, University of Montpellier, 34090 Montpellier, France
[‡]Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM), CNRS UMR 5506, University of Montpellier, 34090 Montpellier, France

Supporting Information

ABSTRACT: Tools to systematically reprogram cellular behavior are crucial to address pressing challenges in manufacturing, environment, or healthcare. Recombinases can very efficiently encode Boolean and history-dependent logic in many species, yet current designs are performed on a case-by-case basis, limiting their scalability and requiring time-consuming optimization. Here we present an automated workflow for designing recombinase logic devices executing Boolean functions. Our theoretical framework uses a reduced library of computational devices distributed into different cellular subpopulations, which are then composed in various manners to implement all desired logic functions at the multicellular level. Our design platform called CALIN (Composable Asynchronous Logic using Integrase Networks) is broadly accessible *via* a web server, taking truth tables as inputs and providing corresponding DNA designs and sequences as outputs (available at http://synbio.cbs.cnrs.fr/calin). We anticipate that this automated design workflow will streamline the implementation of Boolean functions in many organisms and for various applications.



KEYWORDS: *synthetic biology, biological computing, recombinases, logic gates, automated genetic design, distributed multicellular computing*

Reprogramming the response of living cells to chemical or physical signals is a key goal of synthetic biology and would support the development of complex manufacturing processes, sophisticated diagnostics, or cellular therapies.[1] In order to control cellular behavior, researchers have engineered many types of Boolean logic gates operating in single cells by using transcriptional regulators,[2−8] RNA molecules,[9−11] or site-specific recombinases.[12−14] However, scaling-up single-cell logic systems requires solving multiple engineering challenges. First, when program complexity increases (number of inputs ≥3), the high number of parts needed can cause metabolic burden and affect cellular viability. Second, current design methods are mostly *ad-hoc*, and each Boolean function is implemented using a different genetic architecture that needs to be fully characterized and optimized. Despite recent progress toward predictable gate design,[7] some gates simply do not work or are too complex to be implemented within a single cell. Finally, in order to avoid cross-talk, single-cell logic systems need to use different components for every novel signal to be detected. While library of orthogonal regulatory components have greatly expanded,[3,6,15,16] their deployment can be challenging and requires time-consuming optimization.

In nature, division of labor between cellular subpopulations is a ubiquitous mechanism allowing cellular communities to accomplish complex functions.[17,18] Early efforts to engineer synthetic multicellular systems led to the construction of pattern-forming communities,[19] predator−prey ecosystems,[20] synchronized oscillators,[21,22] or distributed metabolic pathways.[23] Researchers also realized that problems faced by logic circuits operating in single cells could be addressed by distributing the logic program between different cells.[24] Because of the spatial separation allowed by cellular compartments, optimized regulatory components can be reused in different subpopulations. As the circuit is divided into smaller subcircuits, metabolic burden is reduced. Finally, simple cellular computing modules can be composed in different manners and wired *via* cell−cell communication channels to obtain different logic functions. For example, Tamsir *et al.* used multilayered circuit designs inspired from electronics to construct all 2-input logic gates by combining spatially separated *E. coli* colonies encoding NOR gates wired *via* quorum-sensing molecules.[25] Specific features of biology can also be used to our advantage to engineer logic systems in a more efficient manner than by strictly transposing electronic designs.[12,24,26] One particularly promising approach is distributed multicellular computation (DMC).[24,27−29] DMC is based on the decomposition of a Boolean function into various subfunctions, each performed by a particular subpopulation of cells. Different subpopulations can then be combined in different manners to realize any given

Received: January 10, 2018
Published: April 11, 2018

Boolean function of interest. Importantly, multiple cells are capable of producing the output which is therefore distributed among the cellular subpopulations. Recently, Macia and colleagues implemented DMC within a multicellular consortium by using cellular computing units performing elementary IDENTITY or NOT operations.[30] While highly scalable, the need for spatial separation between each subpopulation prevents these systems from operating autonomously.

Here we present a composable framework enabling the systematic design of logic gates performing Boolean logic within an autonomous multicellular consortium.

We designed our system to operate using site-specific recombinases, more specifically serine integrases, which allow robust and flexible engineering of complex logic gates.[12,13] Serine integrases are members of the large serine recombinase family[31] and catalyze site-specific recombination between attachment sites *attB* and *attP*. Recombination operates *via* double-strand breaks located at the central dinucleotides followed by the generation of hybrid sites *attL* and *attR*. Depending on the relative orientation of *attB* and *attP*, the recombination reaction leads to excision (parallel orientation) or inversion (antiparallel orientation) of the DNA sequence flanked by the attachment sites.[32] Recombinase devices can implement complex logic functions without the need of cascading multiple logic gates like in electronics.[12,13,26] Integrase recombination is irreversible in the absence of cofactors, so that recombinase logic gates exhibit memory, are single use (one-shot), and therefore belong to the family of asynchronous logic devices (*i.e.*, the system can respond to multiple signals even if they are not present simultaneously).

Our design for Boolean logic is based on a reduced library of cellular computing units responding to one or multiple inputs that can be composed at will to implement all desired Boolean functions (Figure 1). Our logic system is single layer, does not require cell−cell communication nor spatial separation, greatly facilitating its implementation. In order to make our design framework broadly accessible, we provide a fully automated web platform called CALIN (Composable Asynchronous Logic using Integrase Networks) taking truth tables as inputs and providing corresponding DNA designs and sequences as outputs.

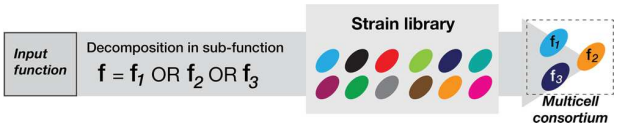


Figure 1. Distribution of a Boolean function within a multicellular consortium. The Boolean function of interest is decomposed as a disjunction (*i.e.*, sum) of subfunctions (or clauses). Here, as an example, a given function, f, is decomposed into functions f_1, f_2, and f_3. The strains performing f_1, f_2 and f_3 are selected from the strain library to assemble a multicellular consortium computing the desired Boolean function.

RESULTS

A Hierarchical Composition Framework for Multicellular Boolean Logic Using Integrase Switches. In order to implement a Boolean function within a multicellular consortium, we decomposed the function into several independent subfunctions, or clauses,[30] executed by a different cellular subpopulation, chosen from a library containing a reduced number of cellular computing units (Figure 1). To facilitate multicellular system composition, we designed our system so that each cellular subpopulation computes independently of the others, without cell−cell communication needed. As a consequence, if one cellular subpopulation is ON (expression of the output gene), the global output of the system is considered to be ON. Because of their reduced number and of the absence of cell−cell communication, cellular computing units can be extensively characterized and optimized to predictably implement all Boolean functions at the multicellular level.

Boolean functions encode the output state of the logic gate. The variables of the function are the inputs of the gate which are equal to 1 if the signal has been present and otherwise to 0. We express Boolean functions using the disjunctive normal form.[33]

The Boolean function f is a disjunction: $f = \beta_1 OR ... OR \beta_M$, where M is the number of clauses present in f, and each β_i is a conjunctive clause: $\beta_i = \theta_{i,1} AND ... AND \theta_{i,n_i}$, where each $\theta_{i,j}$ is a literal of the variable x_j (either the identity of the variable or its negation), with j being an integer between 1 and n_i. n_i corresponds to the number of variables in this conjunction (an integer between 1 and N). N is the number of variables in the function f.

Each cellular computing unit executes a particular "subfunction" corresponding to a conjunctive clause. Then, the full function is performed by combining multiple cellular computing units (Figure 2A).

We designed a hierarchical composition framework in which two elements encoding the NOT and IDENTITY functions (called ID-element and NOT-element) are composed into computational modules which are then combined to generate computational devices executing a particular clause within a cellular subpopulation.

For the sake of simplicity and robustness, we designed switches controlled by integrase-mediated excision (Figure 2B). Excision-based design reduces the distance between gate promoter and the gene of interest. Moreover, as no asymmetric terminator is needed, this design might be easier to deploy into many organisms.[14]

The ID-element consists of a transcriptional terminator flanked by recombination sites and placed between the promoter and the output gene. In presence of the signal, the terminator is excised and the output gene is expressed (Figure 2C, left panel). The NOT-element consists of a promoter driving the output gene and flanked by recombination sites. In presence of the signal, the promoter is excised and the gene is not expressed anymore (Figure 2C, right panel). Computational modules performing conjunctions of NOT or conjunctions of IDENTITY functions are respectively realized by nesting NOT-elements or by placing ID-elements in series (Figure 2D,E). Finally, NOT- and ID-modules are composed in series to obtain the final computational devices: in this case the NOT-module containing the promoter is positioned in 5′ of the ID-module, with the output gene positioned downstream (Figure 2F). Following this hierarchical composition framework, all conjunctive clauses are implementable within a cellular computing unit. The full Boolean function is then executed by a multicellular consortium containing different cellular computing units.

DOI: 10.1021/acssynbio.8b00016
ACS Synth. Biol. 2018, 7, 1406−1412

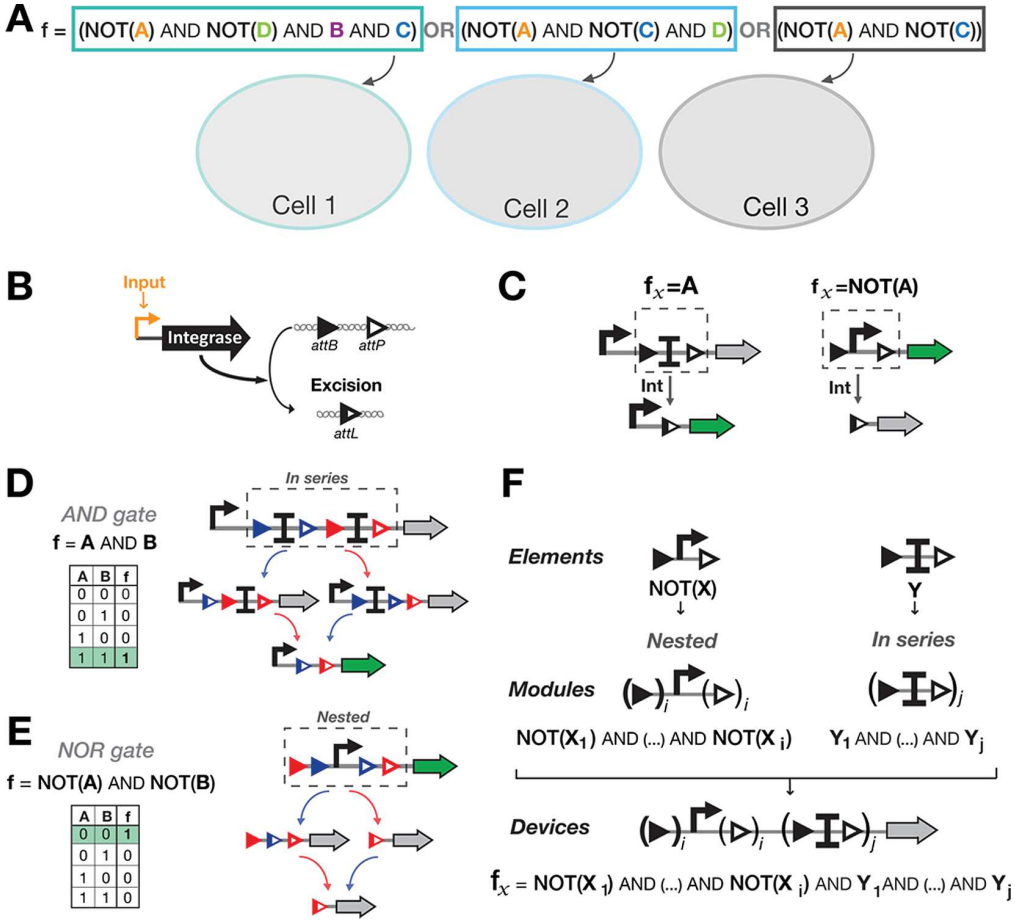


Figure 2. A hierarchical composition framework for asynchronous Boolean recombinase logic. (A) Distribution of a Boolean function within a multicellular consortium by decomposition into conjunctions of literals (variables or their negations). Here an example is depicted in which a Boolean function is decomposed into three subfunctions and implemented in three separate cellular computing units. (B) *attB* and *attP* disposed in parallel orientation. (C) Elements implementing IDENTITY and NOT functions. To obtain an IDENTITY function, a transcriptional terminator is flanked by parallel attachment sites, blocking transcription of the gene of interest. When the signal is present, the terminator is excised and the output gene is expressed. To obtain a NOT function, a promoter is flanked by parallel attachment sites. When the signal is present, the promoter is excised, and the gene is no longer expressed. (D) Functional composition of ID-elements into ID-modules, by placing elements in series to obtain the conjunction of IDENTITY functions. For a 2-input ID-module, the output gene is expressed only when both inputs have been present, both terminators excised (corresponding to an AND gate (*A AND B*)). (E) Functional composition of NOT-elements into NOT-modules, by nesting elements to obtain conjunction of NOT functions. For a 2-input NOT-module, the output gene is expressed only when none of the inputs has been present (corresponding to a NOR gate: *NOT(A) AND NOT(B)*). (F) Hierarchical composition framework for Boolean recombinase logic. ID- and NOT-modules are composed in series, following a priority rule in which the NOT-module is placed upstream the ID-module. The device shown here can be scaled to perform all functions based on conjunction of NOT and IDENTITY functions.

To reduce the number of computational devices, we implemented only one computational device per set of symmetric Boolean functions and interchanged connection between integrases and control signals. For example, the two Boolean functions: *NOT(A) AND B*; *B AND NOT(A)* are executed using the same computational device (Figure S1). Consequently, only 14 computational devices are needed to realize all 4-input Boolean functions (65 536 functions) (Figure 3A). For every additional input (from $N-1$ to N), only $N+1$ novel computational devices are needed while the number of Boolean functions increases drastically. For example, 7 additional devices are needed to transition from 5 to 6 inputs (27 devices in total), enabling a 10^{10} fold increase in the number of Boolean functions (for a total of $\sim 10^{19}$) (Figure 3B). Of note, the different cellular computing units do not always include N integrases and computational devices responding to N inputs. As an example, the 4-input Boolean equation shown in Figure 3D can be executed using 3 strains containing respectively 4, 3, and 2 integrases and with different signal-integrase connectivities.

To implement a N-input Boolean function, a maximum of 2^{N-1} different cellular computing units have to be composed, corresponding to a culture of 2^{N-1} different strains: 4 for 3 inputs and 8 for 4 inputs (Figure 3B). However, most logic functions can be performed using less cellular computing units (an average of 2.3 strains for 3-input and 3.6 strains for 4-input Boolean functions, Figure 3C).

In summary, we provide a hierarchical composition framework using a reduced library of computational devices to systematically implement all N-input Boolean logic functions within a multicellular consortium.

DOI: 10.1021/acssynbio.8b00016
ACS Synth. Biol. 2018, 7, 1406−1412

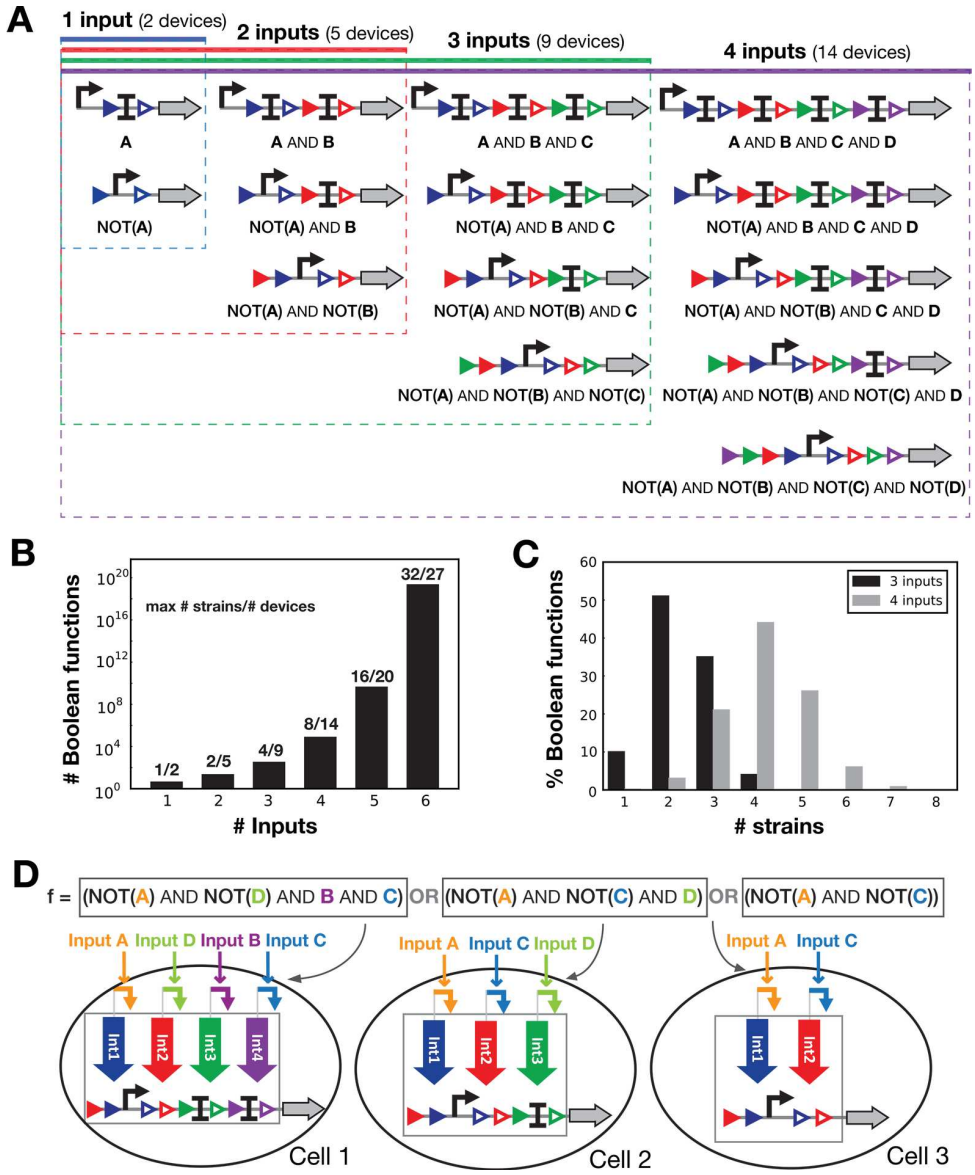


Figure 3. Implementing all Boolean logic functions using a reduced number of computational devices. (A) Schematics of all devices needed to implement up to 4-input functions. (B) Maximum number of strains and number of computational devices needed to compute all Boolean functions for a given number of inputs. See Methods for details. (C) Proportion of Boolean functions implementable with a specific number of strains for 3 and 4 inputs (obtained by generating all the biological designs for 3 and 4-input Boolean functions, see Table S1 for numbers). (D) Example of a biological implementation for a 4-input Boolean function. The function shown here is divided into a disjunction of conjunctive clauses (see Figure 2A). Each conjunctive clause is executed using a particular computational device (defined in panel A) each placed into a separate cellular computing unit. By combining the different units, the full logic function is obtained. If at least one of the cellular units is ON, the output is considered to be ON. Of note, inputs are not always connected to the same integrase (as for input D in Cell 1 and Cell 2), and all integrases and inputs are not present in all cells.

An Automated Design Platform for Recombinase Logic. We then aimed at generating a software for automating the design of cellular consortia performing asynchronous Boolean logic. Softwares enabling such automated genetic circuit design are necessary and extremely useful when the design space becomes too large for humans to explore it efficiently.[7,34−36]

We thus designed an algorithm called CALIN (Composable Asynchronous Logic using Integrase Networks) based on two main steps (Figure 4A). First, the Boolean function of interest is decomposed into a disjunction of conjunctive clauses using the Quine−McCluskey algorithm (see Methods). Then, each clause is converted into a given computational device for which particular connections between integrases and inputs are generated.

The CALIN script written in Python is available on Github and can be directly used for high-throughput generation of biological designs. Furthermore, the CALIN python script can design logic devices customized for specific organisms (*E. coli*, *B. subtilis* and *S. cerevisiae*) and can be tailored by the user to generate devices using fully customized DNA sequences.

DOI: 10.1021/acssynbio.8b00016
ACS Synth. Biol. 2018, 7, 1406−1412

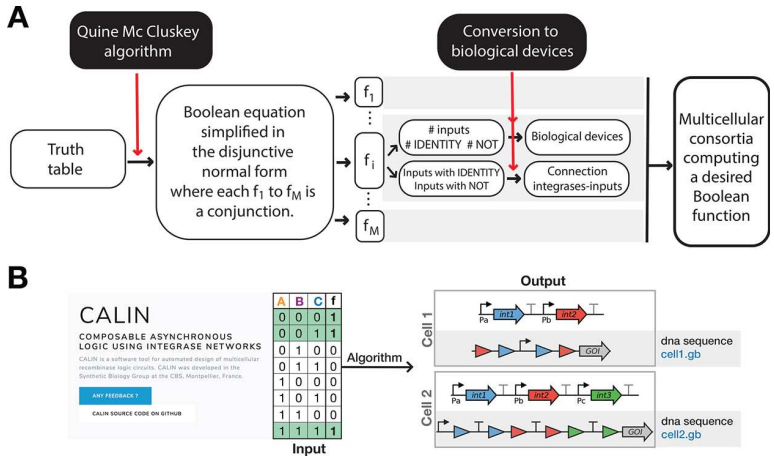


Figure 4. Automated design of multicellular recombinase logic. (A) The CALIN algorithm enables the systematic design of Asynchronous Boolean logic. (B) CALIN web-interface takes as an entry a Boolean truth table and generates as outputs: the connection map between inputs and integrases, the DNA architectures of the computational devices and the corresponding DNA sequences.

In order to enable broader access to our design framework, we also provide a Web site of CALIN accessible at http://synbio.cbs.cnrs.fr/calin.

In the CALIN web-interface, the user fills the number of inputs to process (up to 5) and the desired Boolean truth table or corresponding binary number. The Web site provides as outputs the DNA architectures of the computational devices, the connection map between signals and integrases, and the corresponding DNA sequences (Figure 4B).

DISCUSSION

In this work we present a scalable composition framework for implementing asynchronous Boolean logic within a multicellular consortium. We provide an online design tool for the systematic design of recombinase logic circuits called CALIN (Composable Asynchronous Logic using Integrase Networks). While these designs are currently theoretical, the robustness of integrase-mediated recombination against various site permutations and orientations[12,13,34,37] should support straightforward experimental implementation.

By taking advantage of the single-layer architecture of recombinase logic, we encapsulated complex Boolean functions into various subcellular populations. Because of its compact architecture, our design exhibit two significant improvements over previous DMC systems: (i) no cell−cell communication channels are needed, and (ii) cells do not need to be spatially separated, thereby supporting the implementation of fully autonomous multicellular consortia operating without an external physical device.

Another difference between our system and other DMC is the use of recombinase switches that provide memory.[34,38,39] Recombinase mediated data-storage could be useful for applications requiring endpoint measurements, or delayed readout, like diagnostics. Also, because the state of the logic system is written within DNA, it can be addressed *via* PCR or DNA sequencing,[13,38,40] even if the cells die, providing other robust readout modalities.

As with others DMC systems, for a given number of inputs, the number of elementary computational devices needed to compose all logic functions compares very favorably with the number of possible functions. For example, implementing all 65 536 4-input, or all ~4.3 × 10^9 5-input Boolean functions only requires respectively 14 and 20 computational modules.

As serine recombinases do not require host-specific cofactors and can operate in several species, the designs presented here could be implemented in many organisms. Logicfunctions could also be distributed between different species operating in concert. In such schemes, researchers could take advantage of the particular capacities of different organisms to detect different signals and/or perform specific tasks. Examples of applications include environmental remediation[41,42] or microbiome engineering for therapeutic applications.[43]

A possible challenge for our system is the high number of strains that have to operate together when the number of inputs increases (Figure 3B). Cultivating many strains together could lead to counter selection of some subpopulations, but this problem could be addressed by encapsulating the different strains into hydrogel beads.[40] Also, as the number of strains increases, the output of one subpopulation representing a small fraction of the whole consortia could become difficult to measure. The output level in the ON state will also be different if one or multiple cellular subpopulations are turned ON. However, adding a single cell−cell communication channel could address this problem by propagating the output to the whole-population (Figure S2).

Finally, for some applications, "real-time" response could be achieved *via* a similar composition framework using synchronous recombinase logic gates based on reversible recombination reactions performed by integrases coupled with recombination directionality factors (RDFs) (Figure S3).[12,26]

METHODS

Equations for Determining of Numbers of Functions/Strains/Devices. The number of Boolean functions corresponds to 2 to the power of the number of possible states. As each state can be equal to 1 or to 0, the number of possible states is equal to 2 to the power of N where N is the number of inputs. Consequently, the number of Boolean functions is equal to eq 1.

$$\text{Number}_{\text{Boolean functions}} = 2^{2^N} \quad (1)$$

The maximum number of strains needed to implement any Boolean logic function with N inputs is equal to eq 2, as all N-

DOI: 10.1021/acssynbio.8b00016
ACS Synth. Biol. 2018, 7, 1406−1412

input Boolean equations can be written in the disjunctive normal form, then as a disjunction of a maximum of 2^{N-1} conjunctive clauses.[33]

$$\text{Number}_{\text{strains}} \leq 2^{N-1} \tag{2}$$

The number of different conjunctive clauses (corresponding to a conjunction of literals) is equal to eq 3.

$$\text{Number}_{\text{conjunctive clauses}} = \sum_{k=1}^{N} 2^k \binom{N}{k} \tag{3}$$

If we implement all these functions within cells, the number of standard devices needed is equal to the number of conjunctive clauses (eq 4).

$$\begin{aligned}&\text{Number}_{\text{devices without simplification}}\\ &= \text{Number}_{\text{conjunctive clauses}}\\ &= \sum_{k=1}^{N} 2^k \binom{N}{k}\end{aligned} \tag{4}$$

This method leads to a high number of devices. Therefore, we decided to construct only one device per set of symmetric Boolean functions (*e.g.*, *A AND NOT(B)* is the symmetric function of *NOT(A) AND B*). This approach reduces the number of standard devices. In consequence, for an *N*-input Boolean function, devices computing from 1 to *N* inputs are needed and $k + 1$ nonsymmetric Boolean functions computing the conjunction of *k* literals exist:

$$\text{Number}_{\text{devices}} = \sum_{k=1}^{N} (k + 1) \tag{5}$$

Of note, the number of devices follows the arithmetic series: $\frac{N}{2}(2a_1 + (N - 1)d)$ where $a_1 = 2$, $d = 1$, and *N* is the number of inputs.

In a first approximation, *N* sensor-modules in which a control signal (*i.e.*, a sensor device responding to an input of interest) is connected to an integrase are needed for the construction of an *N*-input system. However, as we reduced the number of devices to a set composed of nonsymmetric Boolean functions, we need to connect all control signals to all integrases to compute all Boolean functions. Therefore, N^2 sensor-modules are needed.

Automated Generation of Genetic Designs. We encoded an algorithm generating genetic designs executing N-input Boolean functions using Python (Figure S4). The algorithm takes as input a Boolean truth table or the binary number corresponding to the function. The output corresponds to the biological implementation of the Boolean function, such as for each strain: a graphical representation of the genetic circuit and its associated DNA sequences.

The truth table is transformed into a Boolean function in the disjunctive normal form using the Quine–McCluskey algorithm[33] (Figure 4A). The Boolean function is decomposed into conjunctive clauses (conjunction of literals). In this scheme, each clause can be regarded as a "subfunction". From each conjunctive clause, we extract two types of information. First, based on the number of IDENTITY and NOT functions, we identify which logic device is needed. Second, based on the association of inputs to either IDENTITY and NOT functions, we identify which sensor-modules are needed among the different connection possibilities between control signals and integrases. Finally, we combine the designs executing the different conjunctive clauses to obtain the global design for implementing the desired truth table.

To simplify the construction process, the DNA sequence of the computational devices is generated by our Python code. In CALIN, sequences are adapted for *E. coli*, but sequence generation can be adapted to other organisms (database available for *B. subtilis* and *Saccharomyces cerevisiae*) or customized using the source Python code available on github.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acssynbio.8b00016.

Figures S1–S3; Table S1 (PDF)

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: jerome.bonnet@inserm.fr.

ORCID

Jerome Bonnet: 0000-0002-8420-9359

Notes

The authors declare no competing financial interest.

The CALIN Web server can be found at http://synbio.cbs.cnrs.fr/calin. CALIN source code is available at https://github.com/sguiz/calin.

■ ACKNOWLEDGMENTS

We thank L. Ciandrini, G. Cambray, G. Labesse, members of the synthetic biology group and of the CBS, P. Lemaire and P. Hersen for fruitful discussions, J. L. Pons and L. Bonnet for help with the CALIN website. Support was provided by an ERC Starting Grant "COMPUCELL", the CNRS/INSERM Atip-Avenir program and the Bettencourt-Schueller Foundation to J.B. S.G. was supported by Ph.D. fellowships from the French Ministry of Research and from the FRM: Fondation pour la Recherche Medicale (FDT20170437282). The CBS acknowledges support from the French Infrastructure for Integrated Structural Biology (FRISBI) ANR-10-INSB-05-01.

■ REFERENCES

(1) Endy, D. (2005) Foundations for engineering biology. *Nature 438*, 449–453.

(2) Fischer, C., and Fussenegger, M. (2004) BioLogic gates enable logical transcription control in mammalian cells. *Biotechnol. Bioeng. 87*, 478.

(3) Stanton, B. C., Nielsen, A. A. K., Tamsir, A., Clancy, K., Peterson, T., and Voigt, C. A. (2014) Genomic mining of prokaryotic repressors for orthogonal logic gates. *Nat. Chem. Biol. 10*, 99–105.

(4) Anderson, J. C., Voigt, C. A., and Arkin, A. P. (2007) Environmental signal integration by a modular AND gate. *Mol. Syst. Biol.*, DOI: 10.1038/msb4100173.

(5) Wang, B., Kitney, R. I., Joly, N., and Buck, M. (2011) Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. *Nat. Commun. 2*, 508.

(6) Moon, T. S., Lou, C., Tamsir, A., Stanton, B. C., and Voigt, C. A. (2012) Genetic programs constructed from layered logic gates in single cells. *Nature 491*, 249–253.

(7) Nielsen, A. A. K., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D., and Voigt, C. A. (2016) Genetic circuit design automation. *Science 352*, aac7341.

DOI: 10.1021/acssynbio.8b00016
ACS Synth. Biol. 2018, 7, 1406–1412

(8) Ausländer, S., Ausländer, D., Müller, M., Wieland, M., and Fussenegger, M. (2012) Programmable single-cell mammalian biocomputers. *Nature 487*, 123−127.
(9) Rinaudo, K., Bleris, L., Maddamsetti, R., Subramanian, S., Weiss, R., and Benenson, Y. (2007) A universal RNAi-based logic evaluator that operates in mammalian cells. *Nat. Biotechnol. 25*, 795−801.
(10) Win, M. N., and Smolke, C. D. (2008) Higher-Order Cellular Information Processing with Synthetic RNA Devices. *Science 322*, 456−460.
(11) Lucks, J. B., Qi, L., Mutalik, V. K., Wang, D., and Arkin, A. P. (2011) Versatile RNA-sensing transcriptional regulators for engineering genetic networks. *Proc. Natl. Acad. Sci. U. S. A. 108*, 8617−8622.
(12) Bonnet, J., Yin, P., Ortiz, M. E., Subsoontorn, P., and Endy, D. (2013) Amplifying genetic logic gates. *Science 340*, 599−603.
(13) Siuti, P., Yazbek, J., and Lu, T. K. (2013) Synthetic circuits integrating logic and memory in living cells. *Nat. Biotechnol. 31*, 448−452.
(14) Weinberg, B. H., Pham, N. T. H., Caraballo, L. D., Lozanoski, T., Engel, A., Bhatia, S., and Wong, W. W. (2017) Large-scale design of robust genetic circuits with multiple inputs and outputs for mammalian cells. *Nat. Biotechnol. 35*, 453−462.
(15) Rhodius, V. A., Segall-Shapiro, T. H., Sharon, B. D., Ghodasara, A., Orlova, E., Tabakh, H., Burkhardt, D. H., Clancy, K., Peterson, T. C., Gross, C. A., and Voigt, C. A. (2013) Design of orthogonal genetic switches based on a crosstalk map of σs, anti-σs, and promoters. *Mol. Syst. Biol. 9*, 702.
(16) Yang, L., Nielsen, A. A. K., Fernandez-Rodriguez, J., McClune, C. J., Laub, M. T., Lu, T. K., and Voigt, C. A. (2014) Permanent genetic memory with > 1-byte capacity. *Nat. Methods 11*, 1261−1266.
(17) Hays, S. G., Patrick, W. G., Ziesack, M., Oxman, N., and Silver, P. A. (2015) Better together: engineering and application of microbial symbioses. *Curr. Opin. Biotechnol. 36*, 40−49.
(18) Wintermute, E. H., and Silver, P. A. (2010) Dynamics in the mixed microbial concourse. *Genes Dev. 24*, 2603−2614.
(19) Basu, S., Gerchman, Y., Collins, C. H., Arnold, F. H., and Weiss, R. (2005) A synthetic multicellular system for programmed pattern formation. *Nature 434*, 1130−1134.
(20) Balagaddé, F. K., Song, H., Ozaki, J., Collins, C. H., Barnet, M., Arnold, F. H., Quake, S. R., and You, L. (2008) A synthetic Escherichia coli predator-prey ecosystem. *Mol. Syst. Biol. 4*, 187.
(21) Danino, T., Mondragón-Palomino, O., Tsimring, L., and Hasty, J. (2010) A synchronized quorum of genetic clocks. *Nature 463*, 326−330.
(22) Prindle, A., Samayoa, P., Razinkov, I., Danino, T., Tsimring, L. S., and Hasty, J. (2012) A sensing array of radically coupled genetic 'biopixels'. *Nature 481*, 39−44.
(23) Shong, J., Jimenez Diaz, M. R., and Collins, C. H. (2012) Towards synthetic microbial consortia for bioprocessing. *Curr. Opin. Biotechnol. 23*, 798−802.
(24) Macía, J., Posas, F., and Solé, R. V. (2012) Distributed computation: the new wave of synthetic biology devices. *Trends Biotechnol. 30*, 342−349.
(25) Tamsir, A., Tabor, J. J., and Voigt, C. a. (2011) Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature 469*, 212−215.
(26) Subsoontorn, P. (2014) Reliable Functional Composition of a Recombinase Device Family, Ph.D. thesis, https://purl.stanford.edu/sm186rb9123.
(27) Regot, S., Macía, J., Conde, N., Furukawa, K., Kjellén, J., Peeters, T., Hohmann, S., de Nadal, E., Posas, F., and Solé, R. (2011) Distributed biological computation with multicellular engineered networks. *Nature 469*, 207−211.
(28) Macia, J., and Sole, R. (2014) How to make a synthetic multicellular computer. *PLoS One 9*, e81248.
(29) Goñi-Moreno, A., Amos, M., and de la Cruz, F. (2013) Multicellular Computing Using Conjugation for Wiring. *PLoS One 8*, e65986.
(30) Macia, J., Manzoni, R., Conde, N., Urrios, A., de Nadal, E., Solé, R., and Posas, F. (2016) Implementation of Complex Biological Logic Circuits Using Spatially Distributed Multicellular Consortia. *PLoS Comput. Biol. 12*, e1004685.
(31) Groth, A. C., and Calos, M. P. (2004) Phage integrases: biology and applications. *J. Mol. Biol. 335*, 667−678.
(32) Grindley, N. D. F., Whiteson, K. L., and Rice, P. A. (2006) Mechanisms of site-specific recombination. *Annu. Rev. Biochem. 75*, 567−605.
(33) Enderton, H., and Enderton, H. B. (2001) *A Mathematical Introduction to Logic*, Academic Press.
(34) Roquet, N., Soleimany, A. P., Ferris, A. C., Aaronson, S., and Lu, T. K. (2016) Synthetic recombinase-based state machines in living cells. *Science 353*, aad8559.
(35) Marchisio, M. A., and Stelling, J. (2011) Automatic design of digital synthetic gene circuits. *PLoS Comput. Biol. 7*, e1001083.
(36) Otero-Muras, I., Henriques, D., and Banga, J. R. (2016) SYNBADm: a tool for optimization-based automated design of synthetic gene circuits. *Bioinformatics 32*, 3360−3362.
(37) Rubens, J. R., Selvaggio, G., and Lu, T. K. (2016) Synthetic mixed-signal computation in living cells. *Nat. Commun. 7*, 11658.
(38) Ham, T. S., Lee, S. K., Keasling, J. D., and Arkin, A. P. (2008) Design and Construction of a Double Inversion Recombination Switch for Heritable Sequential Genetic Memory. *PLoS One 3*, e2815.
(39) Hsiao, V., Hori, Y., Rothemund, P. W., and Murray, R. M. (2016) A population-based temporal logic gate for timing and recording chemical events. *Mol. Syst. Biol. 12*, 869.
(40) Courbet, A., Endy, D., Renard, E., Molina, F., and Bonnet, J. (2015) Detection of pathological biomarkers in human clinical samples via amplifying genetic switches and logic gates. *Sci. Transl. Med. 7*, 289ra83.
(41) Li, L., Yang, C., Lan, W., Xie, S., Qiao, C., and Liu, J. (2008) Removal of methyl parathion from artificial off-gas using a bioreactor containing a constructed microbial consortium. *Environ. Sci. Technol. 42*, 2136−2141.
(42) De Lorenzo, V. (2008) Systems biology approaches to bioremediation. *Curr. Opin. Biotechnol. 19*, 579−589.
(43) Mimee, M., Citorik, R. J., and Lu, T. K. (2016) Microbiome therapeutics - Advances and challenges. *Adv. Drug Delivery Rev. 105*, 44−54.

DOI: 10.1021/acssynbio.8b00016
ACS Synth. Biol. 2018, 7, 1406−1412

2.2 Implementation of multicellular Boolean logic using recombinase switches

Introduction

Since its inception, the field of synthetic biology has been aiming to control ("program") cellular and organismal behavior. To do so, researchers were inspired from electronics and rebuilt devices operating in a similar manner to electronic logic gates but using biological molecules. These "bio"-logic gates operate within the framework of Boolean logic, respond to molecular or physical signals, and most often control gene expression as output.

Implementing Boolean logic circuits within living organisms is of interest for various applications, such as environmental remediation, medical diagnostics, cellular therapeutics, but also for more basic research applications. However, a reduced number of logic systems directed toward an application has been engineered [Courbet 2015a] [Urrios 2018]. Indeed, the implementation of large logic circuits is still limited to experts, due to the complexity of implementation [Nielsen 2016] or to the lack of reusable and well-characterized logic components [Weinberg 2017].

The theoretical framework for the design of Boolean logic circuits detailed previously is based on the composition of logic devices in a multicellular system. By combining different strains containing computational devices chosen from a reduced library, all Boolean functions can be implemented. For example, only 14 logic devices are needed to implement all 65,536 4-input Boolean functions.

Here we aimed at providing a well-characterized, reusable and reliable toolbox of logic devices enabling non-experts to construct, by simple composition of these devices, any Boolean logic circuit. This approach, based on standard parts, can be envisioned for two reasons. First, the extremely reduced number of devices is amenable to deep optimization. Second, logic devices are entirely decoupled from signal detection, so that after optimized these devices are reusable in other circuit designs.

Tailoring will be performed at the integrase layer, by engineering integrase switches responding to the signal of interest.

The logic devices result from the composition of NOT and IDENTITY elements, which are themselves assembled from the following DNA parts: integrase recombination sites, promoters, terminators, 5'UTRs, and gene coding sequences.

Our theoretical framework provides a global design for logic devices but there is still a large degree of freedom to design the corresponding DNA sequence. For example, integrase target sites can be arranged in four different orientations and relative positions. Moreover, various integrases, terminators, promoters, and translational control elements can be used.

To translate a given theoretical design into a DNA sequence encoding a logic device, all these parameters need to be defined.

Our goal was to engineer well-characterized, reusable, and reliable logic devices with a consistent and precise behaviors. The logic devices must fit as much as possible to the binary states of the Boolean equation. Therefore, gene expression from the output gene should have a low background level in the OFF state and a high fold change between ON and OFF states. Additionally, constant OFF and ON states between the different input states and logic devices is highly desirable. This level matching is required for composition of the devices in a multicellular system.

In this work, we engineered a collection of logic devices to implement logic functions with up to four inputs with the synthetic biology bacterial workhorse, *Escherichia coli*.

We took a bottom-up approach with the goal to probe, and possibly achieve the functional composition of basic parts into computational elements and then into computational devices. We first defined the identity of each part: integrase sites, promoter, terminators, 5'UTR, and gene. We built and characterized all possible versions of computational elements (NOT and ID-gates) for four integrases. We then selected elements that best fit the expected behavior and assembled these elements into computational devices, which were subsequently composed in multicellular logic systems. We obtained the 14 logic devices behaving as expected without optimization. The common threshold between these devices will permit the implementation of complex logic function in multi-cellular systems. In parallel, we characterized individual parts and compared the predicted behavior of elements resulting from different part compositions with experimental measurements. The part characterizations show high effects on transcription efficiency of integrase sites.

2.2.1 Selection of a set of four orthogonal integrases

We selected four orthogonal integrases to construct up to 4-input logic devices. We selected well-behaved Bxb1 and TP901 integrases previously used to engineer 2-input logic gates [Bonnet 2013]. Yang and colleagues identified and characterized a set of 14 serine integrases [Yang 2014]. From this set of integrases, we selected four orthogonal ones: Int3, Int4, Int5 and Int7 as the ones with the most efficient recombination rates according to the authors. We then tested the orthogonality of Int3, Int4, Int5, Int7, Bxb1 and Tp901 integrases. To do so, for each integrase, we built a BP target and its corresponding LR target [Bonnet 2012]. BP targets are composed of a constitutive promoter surrounded by attB and attP integrase sites in antiparallel orientation (inversion mode). A different gene expression cassette is positioned on each side. Therefore, in absence of integrase, one output gene is expressed (either GFP or RFP). In presence of integrase, the promoter is inverted, and the construct switches from expressing one gene to the other. For Bxb1 and Tp901 integrases, BP target constructs are

from [Bonnet 2012, Bonnet 2013] and switches from GFP to RFP expression upon integrase recombination (Figure 2.1A). I built BP targets for Int3, Int4, Int5 and Int7, that switch from RFP to GFP expression (Figure 2.1B). This might be an illustrative example of the contrarian behavior most Ph.D. students have with their Ph.D. supervisor.

We tested the activity of the six integrases by co-transforming vectors containing the BP targets with vectors expressing the corresponding integrase. For the expressions of Bxb1 and Tp901 integrases, we used the dual controller from Bonnet et al. [Bonnet 2013], in which the expression of Bxb1 integrase is induced by aTc and expression of Tp901 integrase by arabinose. For Int3, Int4, Int5, and Int7, we cloned each integrase under a Plac promoter inducible by IPTG (more details on the construction can be found in the Material and Methods section). After co-transformations of the target with the integrase vectors and induction of integrase expression, we measured GFP and RFP expression levels via flow cytometry and compared them to strains containing the BP target alone. All integrases triggered a clear switch in gene expression when expressed in the presence of their cognate target (Figure 2.1C). However, for Tp901 and Int3, the differences of gene expression between the two states were not clear. For Tp901 integrase, a low but significant level of GFP was still produced after switch and the same behavior was observed for Int3 for RFP. From part characterization data, we hypothesized that this residual gene expression is due to cryptic promoter activities of the integrase sites.

To test the orthogonality of this set of integrases, we co-transformed, pair by pair, all BP target vectors with all integrase vectors. We quantified the percentage of switch for each BP target/integrase combination via flow-cytometery. We plotted GFP fluorescence over RFP fluorescence intensities and defined gates corresponding to the BP target alone or the LR target. The percentages of bacteria in BP or LR state are displayed in the two heatmaps in Figure 2.1D. For all BP targets with the corresponding integrases, we obtained almost 100% of the population in LR state and almost 0% in BP state, as shown previously in Figure 2.1C. For BP targets co-transformed with a non-cognate integrase, we obtained almost 100% of the population in BP state and almost 0% in LR state, except for Tp901 integrase combined with the BP target of Int3. For this particular combination, we obtained a mixed population which does not correspond to GFP and RFP expression levels of the BP target neither the LR target. Therefore, except Tp901 with BP target of Int3, integrases are specific to their BP targets and are orthogonal.

We finally selected Bxb1, Tp901, Int5, and Int7 to build our logic devices. As according to our results Int3 is not compatible with Tp901, we chose to keep Tp901 instead of Int3 because we already had several constructs using Tp901. Moreover, based on [Yang 2014], Int3 and Int4 integrases are toxic for the cells at high expression level (Figure 2d of [Yang 2014]).

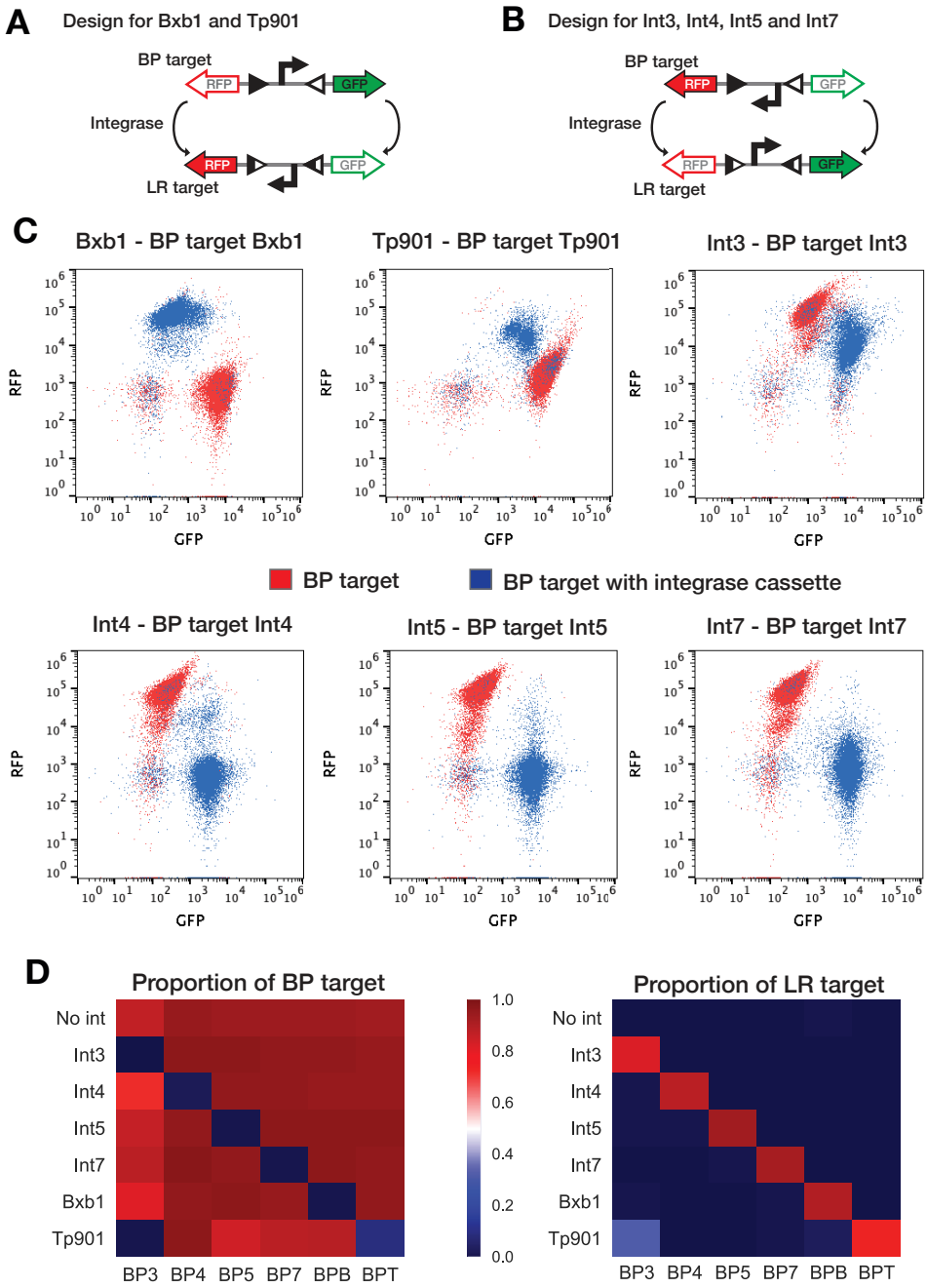


Figure 2.1: **Orthogonality of a set of 6 serine integrases.** (A) and (B): Design of BP targets for the 6 integrases. (A) For Bxb1 and Tp901 integrase, in presence of the integrase, gene expression switches from GFP to RFP via promoter inversion (B) for Int3, Int4, Int5 and Int7, gene expression switches from RFP to GFP. (C) Characterization of each integrases via co-transformation with BP targets. The graphs correspond to the density plots of flow-cytometer experiments with GFP over RFP fluorescence intensity in arbitrary unit. The red dots are *E. coli* strains with BP targets and the blue dots are *E. coli* co-transformed with BP targets and corresponding integrase cassettes. Cells are grown overnight in LB, and with the corresponding inducers for the expression of integrases. (D) Heatmaps of the proportion of BP target (left side) and LR target (right side) in the population of bacteria measured by flow-cytometer. For both heatmaps, each square corresponds to a co-transformation of one integrase cassette or none (labeled in y axis) with one BP target (labeled in x axis, with BP3 for BP target of Int3, BP4 for Int4, BP5 for Int5, BP7 for Int7, BPB for Bxb1 integrase and BPT for Tp901 integrase).

2.2.2 Design of a standard logic device architecture

We designed a shared standard logic device architecture for all constructs (Figure 2.2A). This architecture generates standard genetic context for all constructs and facilitates rapid DNA assembly. The cassette is composed of 40 bp spacers to allow assembly of parts via Gibson assembly. The 5'UTR sequence is composed of a ribozyme (RiboJ) [Lou 2012] and a bicistronic RBS (BCD2) [Mutalik 2013b]. The ribozyme catalyzes the cleavage of the mRNA at this position, and therefore insulate the translation from potential secondary structure due to the logic device. Consequently, the sequence of the messenger RNA encoding the output gene is identical for all constructs. The bicistronic RBS is composed of two ribosome binding sites, the first enabling unfolding of the RNA using the helicase activity of the ribosome, and the second mediating the translation of the gene. We used as output gene a superfolder GFP. We placed the logic device upstream of the 5'UTR of the gene. It is composed of integrase sites, a promoter, or a terminator (Figure 2.2B). For our promoter, we used the strong P7 promoter ([Mutalik 2013b]. As terminators, we used terminators from a library (Table 1 in Material and Methods) selected from Chen et al. [Chen 2013]. We chose the terminators with the strongest average strength and highest sequence divergence to avoid unwanted recombination

Ideally, the logic device should be chromosomally integrated, to reduce the load to the cell and for stability of the system. However, due to the large number of constructions that we had to assemble and test, we used a low copy plasmid (pSB4K5) with pSC101 origin of replication and kanamycin antibiotic resistance.

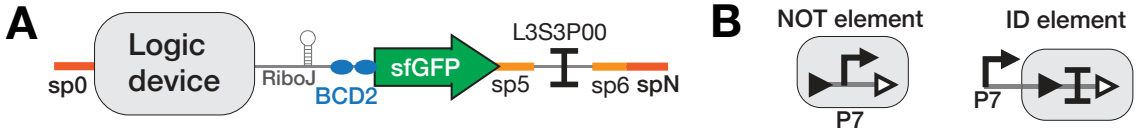


Figure 2.2: **Design of a standard logic device cassette.** A - The standard logic device cassette is composed of 40 bp spacers (sp0, sp5, sp6, and spN) to facilitate cloning by Gibson assembly, a ribozyme (RiboJ), and a bicistron (BCD2) in the 5' end of the output gene, superfolder GFP. Logic devices are placed between the spacer 0 and the ribozyme. B - General design of NOT and ID elements, composed of the P7 promoter surrounded by integrase site in excision orientation for the NOT element, and the P7 promoter followed by a terminator surrounded by integrase sites in excision orientation for the ID element.

2.2.3 Characterization of a set of logic elements

Logic devices are generated by composing NOT and IDENTITY elements. Therefore, we first constructed and characterized a collection of logic elements responding to the 4 integrases selected. NOT-elements are composed of a promoter surrounded by integrase sites, and IDENTITY-elements are composed of a terminator surrounded by integrase sites and a promoter in 5' (Figure 2.2). The two integrase sites are oriented in the same orientation to mediate excision; therefore, sites can be positioned in four different configuratoins. (Figure 2.3A and 2.4A). Due to previous characterization of integrases and to data from Bonnet and colleagues [Bonnet 2013], we supposed that integrase site positions and orientations have an effect on gate behavior. We tested all combinations of integrase sites for each element and each integrase for a total of 32 constructs (2 types elements X 4 possible combinations X 4 integrases). We built 32 constructs corresponding to the DNA sequence resulting from the excision reaction, such as the attL or attR site alone for NOT-element, and the P7 promoter and attL or attR integrase site for IDENTITY element. We aimed at avoiding sequence repetitions in our constructs, which can lead to sequence instability [Nielsen 2016]. Thus, for ID-elements, we used different terminators for each integrase. Consequently, the behaviors of these IDENTITY-elements are highly dependent on the selected terminators (Table 1 - Material and Methods).

We then designed, synthesized and cloned these 64 constructs using a standard workflow detailed in Materials and Methods.

We characterized all constructs using a flow cytometer. Cells were grown overnight in 96 well-plates in LB after which GFP fluorescent intensity was measured. Results are represented in Figure 2.3B and 2.4B in fold change relative to the negative control, the standard logic device cassette without any insert (i.e. no promoter). The positive control corresponds to the P7 promoter driving expression of GFP. We observed important differences in fluorescence intensity between the different orientations of integrase sites, in particular for the ID-elements.

For NOT elements, we expected the non-recombined constructs to express GFP at a similar level to the positive control; for the sequences corresponding to elements after excision (i.e. attL or attR sites), as no promoter is present, we expected fluorescence similar to the negative control. For all NOT elements, we observed GFP expression levels lower than for the positive control, from 1.2 to 5 times lower (Figure 2.3). This decrease of gene expression is probably due to transcriptional attenuation mediated by the integrase sites positioned between the promoter and the 5'UTR of the gene. At least two mechanisms can explain this transcriptional attenuation. First, the additional DNA sequence increases the distance between the site of transcription initiation and the gene coding sequence; previous work showed that such increase tends to decrease transcription efficiency [Chizzolini 2014]. Secondly, the integrase sites may form secondary structures which therefore decrease transcription efficiency. We hypothesized that the second mechanism is more relevant, because the sites are short (~40-50 bp) and have semi-palindromic sequences that are prone to secondary structure formation. Indeed, accord-

ing to the following site characterization, Bxb1 attB and attP sites induce a decrease in gene expression, corresponding to decrease of gene expression for the NOT BF-PF and NOT PF-BF elements.

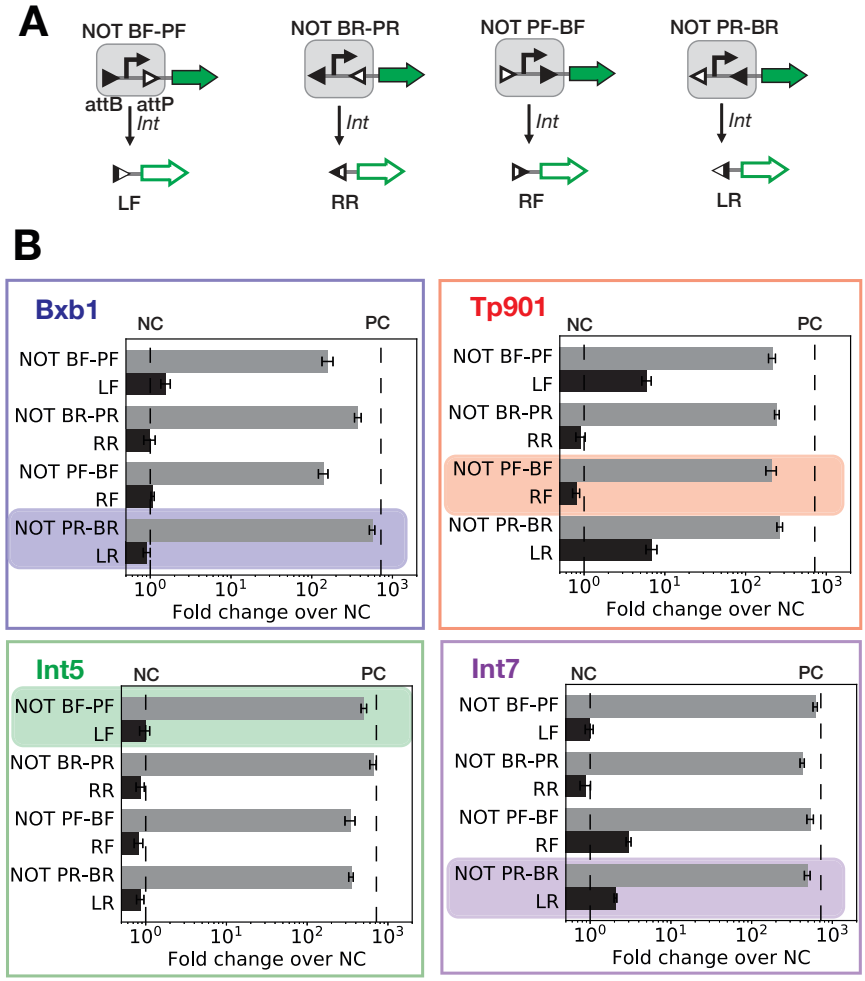


Figure 2.3: **Characterization of a library of NOT-elements.** (A) The 4 possible designs of NOT-elements, with a promoter flanked by integrase sites, and the corresponding constructs after excision mediated by the integrase. Triangles correspond to the integrase sites, attB site in black, attP site in white, and attL or attR in black and white. F denotes sites in forward orientation and R for reverse. The gene coding sequence is a superfolder GFP and the promoter is the P7 promoter. (B) Bar graphs correspond to the fold change in mean fluorescence intensity of constructs compared to the negative control. Data were obtained by flow-cytometry measurement using 3 replicates per experiments, from 3 experiments performed on different days. The grey bars correspond to NOT-elements and the black bars to the attL and attR sites resulting from integrase-mediated excision. The dash lines correspond to fold change of the negative control (NC, equal to 1) and the fold change of the positive control (construct with only promoter P7). Error bars represent the mean of the standard deviation between the three replicates in each experiment.

For Bxb1 integrase, we observed a clear difference in gene expression between elements with the sites in forward orientation (gene expression around 4 times lower than positive control) and the ones in reverse orientation (gene expression around 1.2-1.8 times lower than positive control).

For others integrases, expression levels are similar between the different NOT-elements.

For attL and attR integrase sites alone, corresponding to the element after excision, the GFP fluorescence intensity is comparable to the negative control (i.e. background autofluorescence) except for attL Tp901 integrase sites, attL Int7 site in reverse, and attR Int7 site in forward orientations. AttL Tp901 integrase site in reverse and forward orientation behave as a low efficiency promoter with a GFP expression of around 6 times above the negative control. For Int7, attL reverse, and attR forward exhibit a 2- to 3-fold change of expression over the negative control.

We selected NOT-elements with the most binary behavior, such as an expression level close to the positive control for the element in absence of integrases and close to the negative control after excision.

For ID-element, we expected constructs to show low GFP fluorescence intensity in the absence of integrase (ID-element construct) and to express GFP in the presence of integrase, after excision of the terminator (attL and attR constructs). Results are presented in Figure 2.4B. Because we used a strong promoter, despite having chosen strong terminators most of ID-elements expressed a significant level of GFP, even without integrase. Moreover, for all integrases, important differences between integrase site orientations were observed, with up to 100-fold differences between various arrangements of Int5 elements. Tp901 integrase ID-elements have the highest gene expression leakages, which could be due to the low efficiency of the terminator used, the interactions between the terminator and the integrase sites, or the cryptic promoter activities of the integrase sites. A more precise individual characterization of integrase sites and terminators was later performed later. The leakage seems due to the terminator according the characterization of an element with a different terminator, however no clear results were obtained as we did not succeed to clone the T2 terminators in front of P7 promoter for characterization.

For most of the sequences corresponding to excised ID-elements, GFP expression levels were similar to the expression level of the positive control, as expected. However, some constructs exhibited expression levels lower than the positive control. Again, for a given integrase, we observed important differences between integrase site orientations. For example, the construct composed of the P7 promoter and attL Bxb1 site in forward orientation had an expression level 4 times lower than the similar construct with attR forward.

After characterization, we selected one ID-element per integrase (Figure2.4B). For Tp901, we found that the terminator used initially was not efficient enough, leading to high leakage in the OFF state. We thus switched the T2 terminator (ECK120029600) to B0015 and obtained an ID element with low leakage and high dynamic range (Figure 2.4C).

In conclusion, we characterized all possible element architectures and identified elements with a specific parts arrangement for each integrase that produced the desired behavior. We then composed those into higher order computational devices. Importantly, we show that

integrase sites can have important effect on transcription, at least in *E. coli*. We also found that att sites can have directional terminator or promoter activities. These effects on transcriptional output must be measured and taken into account when designing recombinase devices.

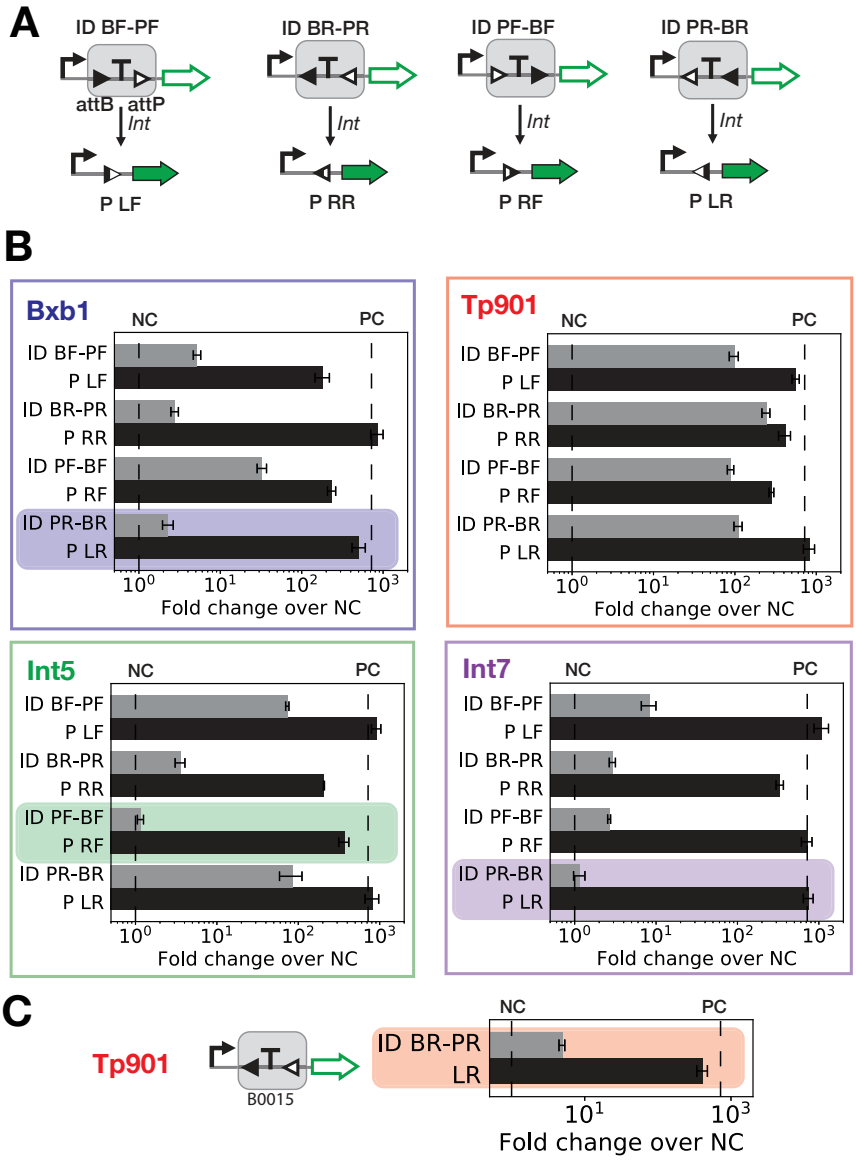


Figure 2.4: **Characterization of a library of ID-elements.** (A) The four possible designs of ID-elements with a terminator flanked by an integrase site pair, and the corresponding constructs after integrase-mediated excision. The gene coding sequence is a superfolder GFP and the promoter, the P7 promoter. (B) and (C) Bar graphs correspond to the fold change of mean of fluorescence intensity compared to the negative control. Data were obtained by flow-cytometry measurement with three replicates per experiments, from three experiments performed on different days. The grey bars correspond to ID-elements and the black bars for the attL and attR sites resulting from integrase-mediated excision. The dash lines correspond to fold change of the negative control (NC, equal to 1) and the fold change of the positive control (construct with only promoter P7). Error bars represent the mean of the standard deviation between the three replicates in each experiments. The bars surrounded by a colored box correspond to elements that were selected for assembling computational devices.

2.2.4 Construction and characterization of the 14 computational devices for 4-input multicellular Boolean logic

In order to design the 14 computational devices required for 4-input Boolean logic, we composed the selected NOT- and ID- elements previously characterized (Figure 2.5). We added 20 bp spacers between elements to limit interactions between them and facilitate further modifications. We built all devices following the framework detailed in Material and Methods.

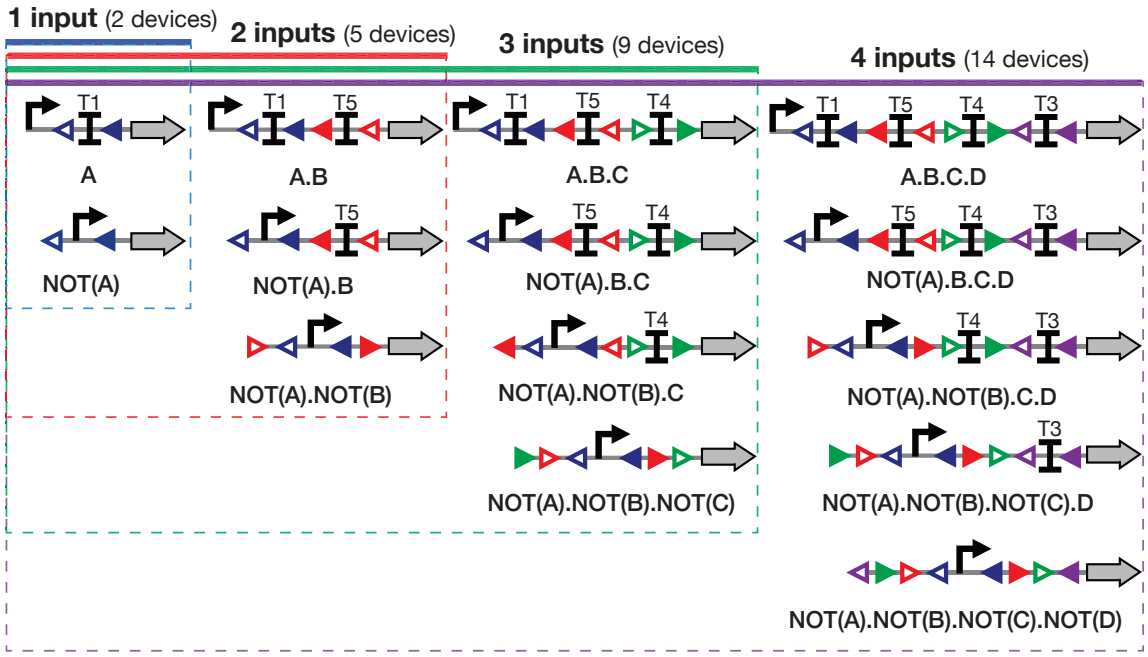


Figure 2.5: **Design of the 14 computational devices based on refined NOT- and ID-elements.** The filled triangles correspond to attB sites, and the empty triangles to attP sites. Blue: Bxb1, red: Tp901, green: Int5, purple: Int7. Arrows are for the promoter P7 and the short name of each terminator is mentioned at the top of them. The output gene is superfolder green fluorescent protein. The Boolean equation implemented by each device is written below the device design.

2.2.4.1 Design and characterization of a combinatorial collection of constitutive integrase generators

We then aimed at characterizing the different states of the computational devices in response to the expression of one or multiple integrases. To do so in a streamlined manner, we built a collection of integrase generators constitutively expressing all possible combinations of the four integrases chosen. We obtained 16 vectors corresponding to the 16 input states of a 4-input truth table (four vectors expressing one integrase, six vectors expressing two integrases, four vectors expressing three integrases, and one vector expressing the four integrases).

To construct these cassettes, we first constructed a template composed of 40 bp spacers for Gibson assembly, constitutive promoters, ribosome binding sites and terminators. Then, we

inserted integrases, each one at a specific locus. We chose promoters and ribosome binding sites described to have intermediate activities so that integrase expression level should be sufficient to have a complete switch while avoiding cellular toxicity described for some integrases [Yang 2014]. Furthermore, we chose promoters with divergent DNA sequences to limit instability of the construction due to sequence repetitions (Figure 2.6A). For the Bxb1 integrase cassette, we selected the strong P5 promoter [Mutalik 2013b] and RBS B0034 (parts.igem.org) as no toxicity of Bxb1 have been observed previously. For Tp901 integrase cassette, we selected the P2 promoter [Mutalik 2013b] and B0032 ribosome binding site (parts.igem.org) of medium efficiency. For Int5, we chose the strong J23100 promoter (parts.igem.org) and the medium RBS-Int5 [Yang 2014] previously optimized for this enzyme, because like Bxb1 integrase, Int5 does not seems to be toxic at high expression level [Yang 2014]. On the other hand, Int7 was described to be harmful for cell growth at high-expression levels. We therefore chose a medium promoter and ribosome binding site for this enzyme (ProC [Davis 2011] and RBS-Int7 [Yang 2014]).

We first constructed each single-integrase cassette separately by using Gibson assembly for insertion of integrase in the template (Figure 2.6B). Then, we tested each of the four single-integrase cassettes by co-transformation with the corresponding BP target. We compared fluorescence intensity profiles with the one of BP target alone. For all single-integrase constructs, a clear switch of gene expression was observed (Figure 2.6C). We constructed and characterized the 12 remaining constructs in a similar manner. The four-integrase construct shows a behavior similar to the single-integrase constructs, demonstrating that all integrases can be expressed constitutively in a single cell without affecting integrase switches efficiency (Figure 2.6D). We performed a full characterization of the vectors and found that all vectors can switch all BP targets corresponding to the integrases they express (Figure 2.6E).

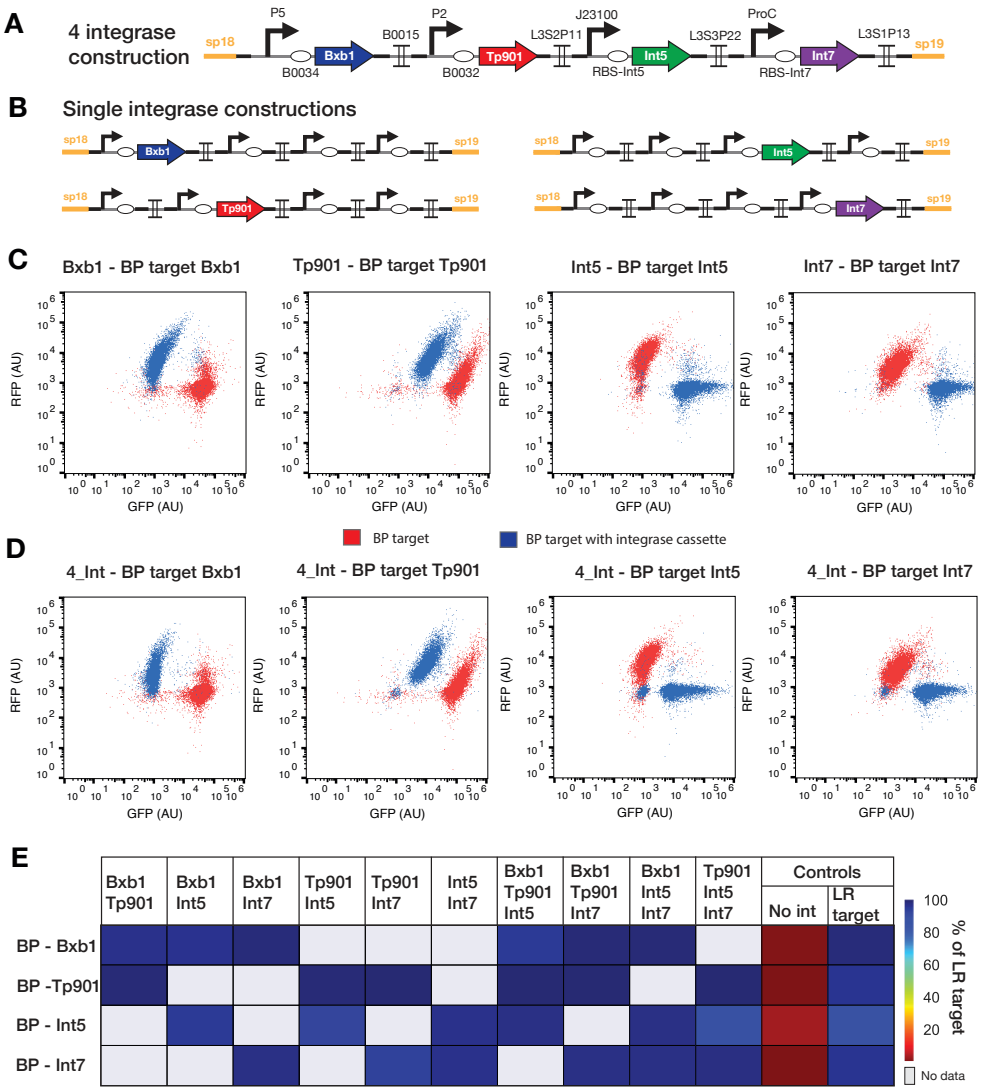


Figure 2.6: **16 constitutive integrase cassettes mediating all 4-input truth table states.** (A) The 4-integrase cassette is composed of the four integrase genes, a different ribosome binding site and promoter for each integrase, terminators to insulate each gene expression cassette and spacers to facilitate gibson assembly cloning. (B) The four single-integrase cassettes. (C) Characterization of the four single-integrase cassettes and (D) of the four-integrase cassette. Each single-integrase cassette was transformed with its BP target and the four-integrase cassette with each BP target. Aafter overnight culture, GFP and RFP fluorescence intensities were measured via flow-cytometery. The graphs correspond to density plots of the bacteria population where the x-axis represents GFP fluorescence intensity in A.U. and the y-axis represents RFP fluorescence intensity in A.U. The dots in blue correspond to the BP target with the corresponding integrase cassette and the dots in red correspond to the BP target alone as negative control of the switch. (E) Characterization of the two- and three-integrase cassettes. Each cassette was transformed with BP targets corresponding to the integrase under which it mediates expression, and after overnight culture, GFP and RFP fluorescence intensities were measured via flow-cytometery. For each target, on the GFP vs. RFP density plot, a gate was defined on the switched population based on the LR target. The percentage of switched population is represented in the heatmap, data represented correspond to the mean of 3 replicates in one experiment. Each square corresponds to one BP target (labeled in the y-axis) and one integrase cassette (corresponding integrase labeled in the x-axis). The grey squares correspond to no conditions were no experiment was performed as the integrase corresponding to the BP target is not present in the integrase cassette.

2.2.4.2 Characterization of computational devices behavior

Using this library of constitutive integrase cassettes, we transformed each logic device with the set of integrase cassettes corresponding to its number of inputs (Figure 2.7). For instance, to test 2-input logic devices, only four integrase cassettes are required corresponding to the four truth table input states (no integrase, Bxb1, Tp901 and Bxb1+Tp901). Similarly, 3-input logic devices are characterized using 8 integrase cassettes and 4-input logic devices using 16 integrase cassettes.

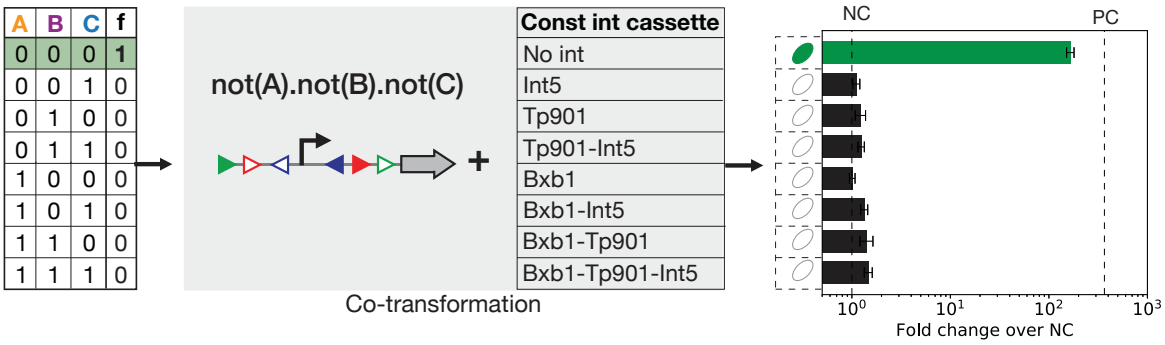


Figure 2.7: **Characterization of a logic device by co-transformation with a constitutive integrase cassette for each input state.**

We found that of all 14 logic devices behaved as expected (Figure 2.8 and 2.9), we measured a good fold change from 30- to 300-fold between off and on states (Figure 2.10B). Despite the variability in fold change observed across devices, they all produce very distinctive ON and OFF states.

In some OFF states, we measured an expression level of GFP above the background, potentially due to leakage of terminators or promoter activities of integrase sites. We quantified the difference between supposedly OFF states and the negative control cells, and termed this difference "Error OFF". We found that the average value of the Error OFF was of approximately 2-fold, with a maximum of 10-fold (Figure 2.10C).

Similarly, in most logic devices, we observed that supposedly ON states had a lower fluorescence intensity than the positive control. We defined this difference as "Error ON". Error ON was highly variable between logic devices, probably because the very different sequences placed between the promoter and the GFP produce highly variable transcriptional attenuation effects (Figure 2.10D). These transcriptional attenuation effects, already observed in isolated elements, are likely to be cumulative when elements are concatenated.

For our logic devices, our results are better than previously published logic devices. Bonnet et al. 2-input AND and NOR gates have a 44- and 33-fold change, and ours a 59- and 90-fold change respectively. For the logic gates of Nielsen et al., it is difficult to estimate as no raw data is available, but NAND fold change seems to be lower than 10.

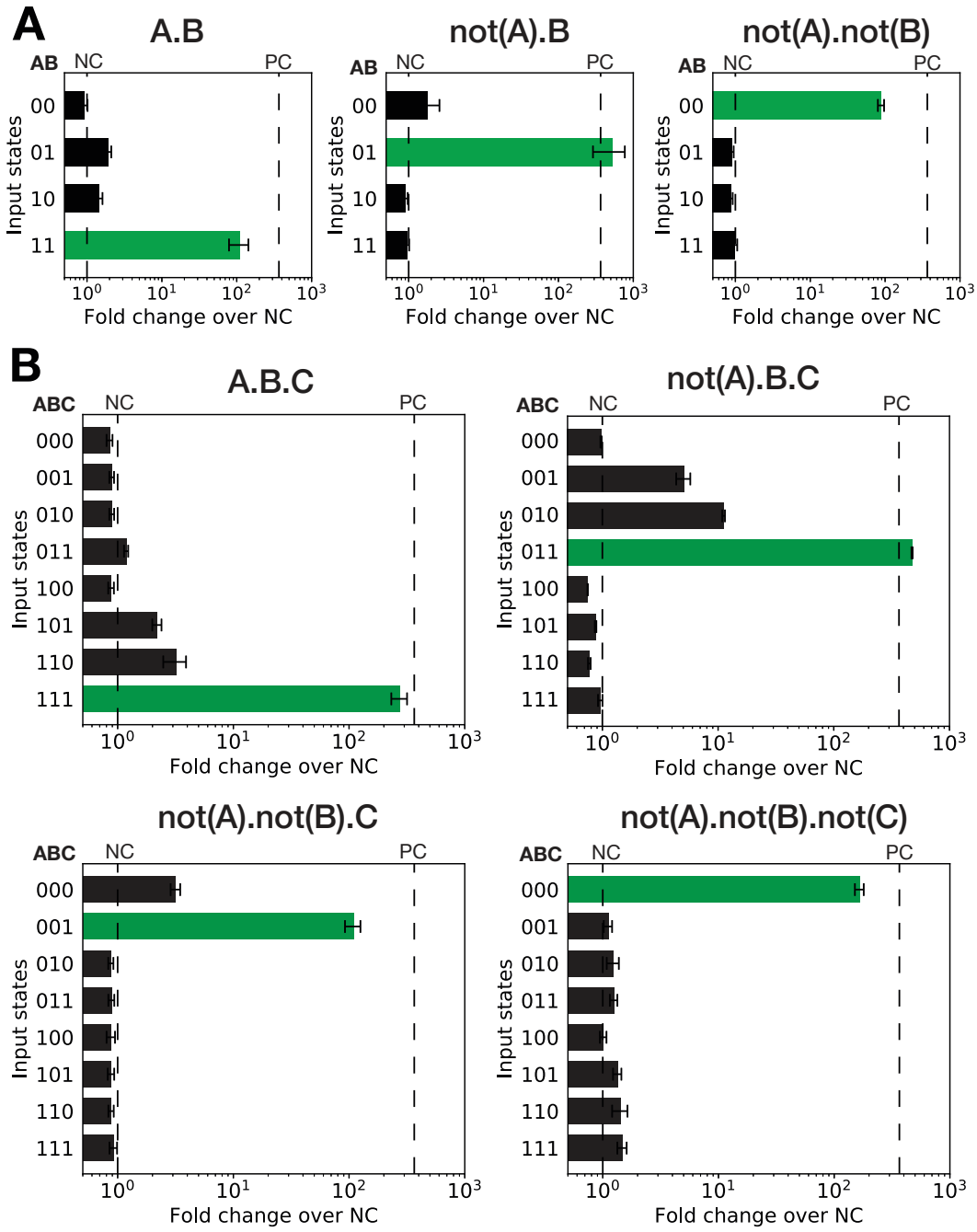


Figure 2.8: **2- and 3-input logic devices characterized in the 4 and 8 truth table states.** Each graph corresponds to the characterization of one logic device in all input states conditions by flow-cytometry experiments measuring GFP fluorescence intensity. Each input states correspond in the experiment to the co-transformation of the logic device plasmid with the constitutive integrase cassette corresponding to the input state. A is for Bxb1 integrase, B for Tp901 integrase and C for Int5 integrase, 1 corresponds to the expression of the corresponding integrase and 0 to its absence. Fold changes over NC correspond to the ratio of the mean fluorescence intensity of two experiments with three replicates per experiment over the mean fluorescence intensity of the negative control which is a GFP cassette without promoter. See Materials and Methods for calculation of the mean fluorescence intensity and the error bars. NC and PC dash lines are negative and positive control fold changes.

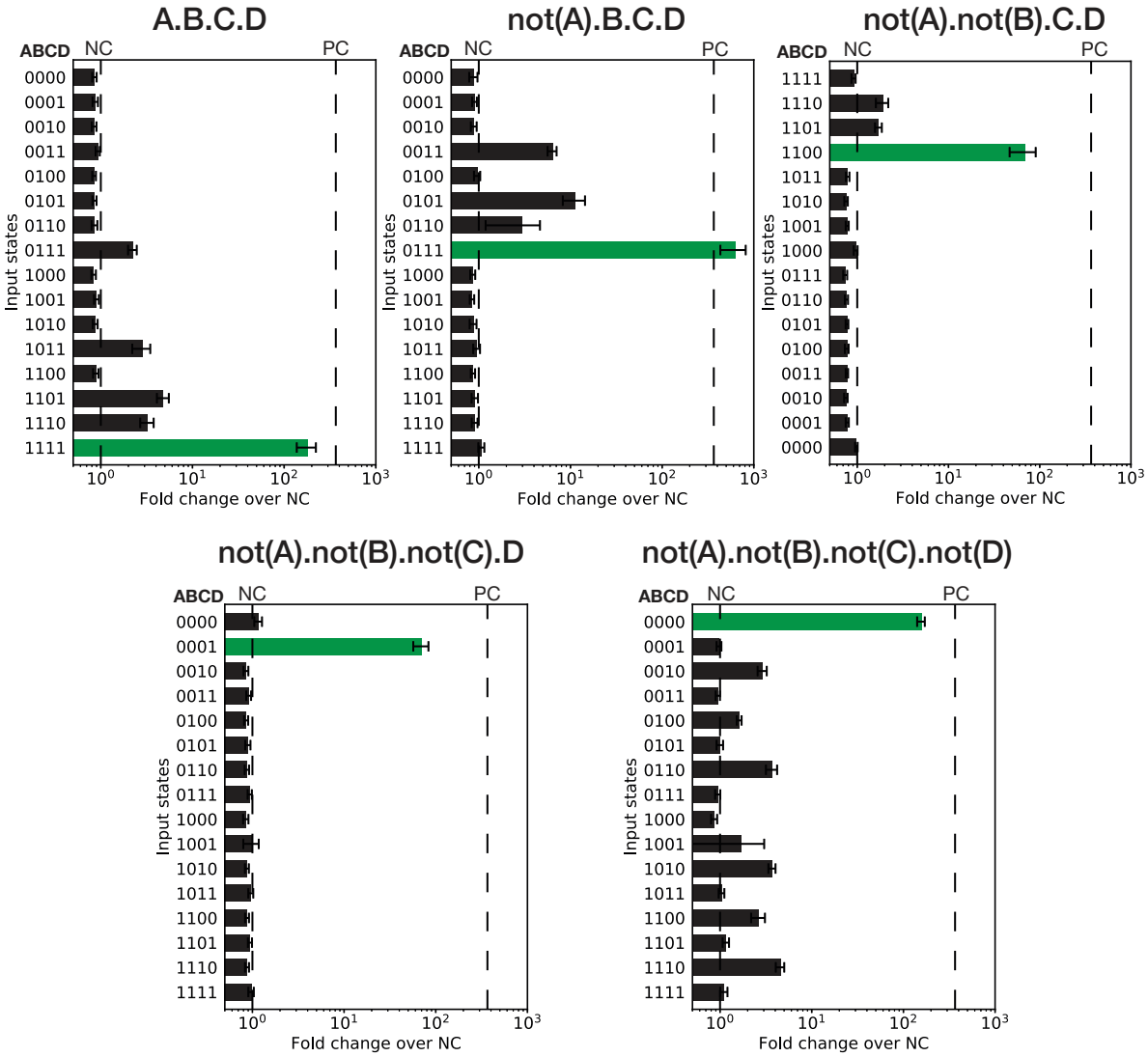


Figure 2.9: **4-input logic devices characterization in the 16 truth table states.** (As for Figure 2.8)

Moreover, as our logic devices are designed to be used in a multicellular consortia system, it required a common OFF and ON threshold between devices. In all devices, the lowest ON state and higher OFF state differ by 6 fold change (corresponding to the ON state of not(A).not(B).not(C).D and OFF state of not(A).B.C.D). This common threshold is better than previously characterized sets of logic devices, as for Nielsen et al. no common threshold exists (for example, A NIMPLY B OFF state is higher than NAND ON state).

However, these logic devices could still be further optimized. In particular we observed a high and variable background expression level in the OFF states. This background level could be critical for operating a multicellular logic system based on our current theoretical framework as we consider that the system output is ON when at least one of the strains is ON. One possible failure mode resulting from leakage problems would be to have multiple strains in a leaky OFF state producing the same global GFP fluorescence intensity as one strain in an ON state.

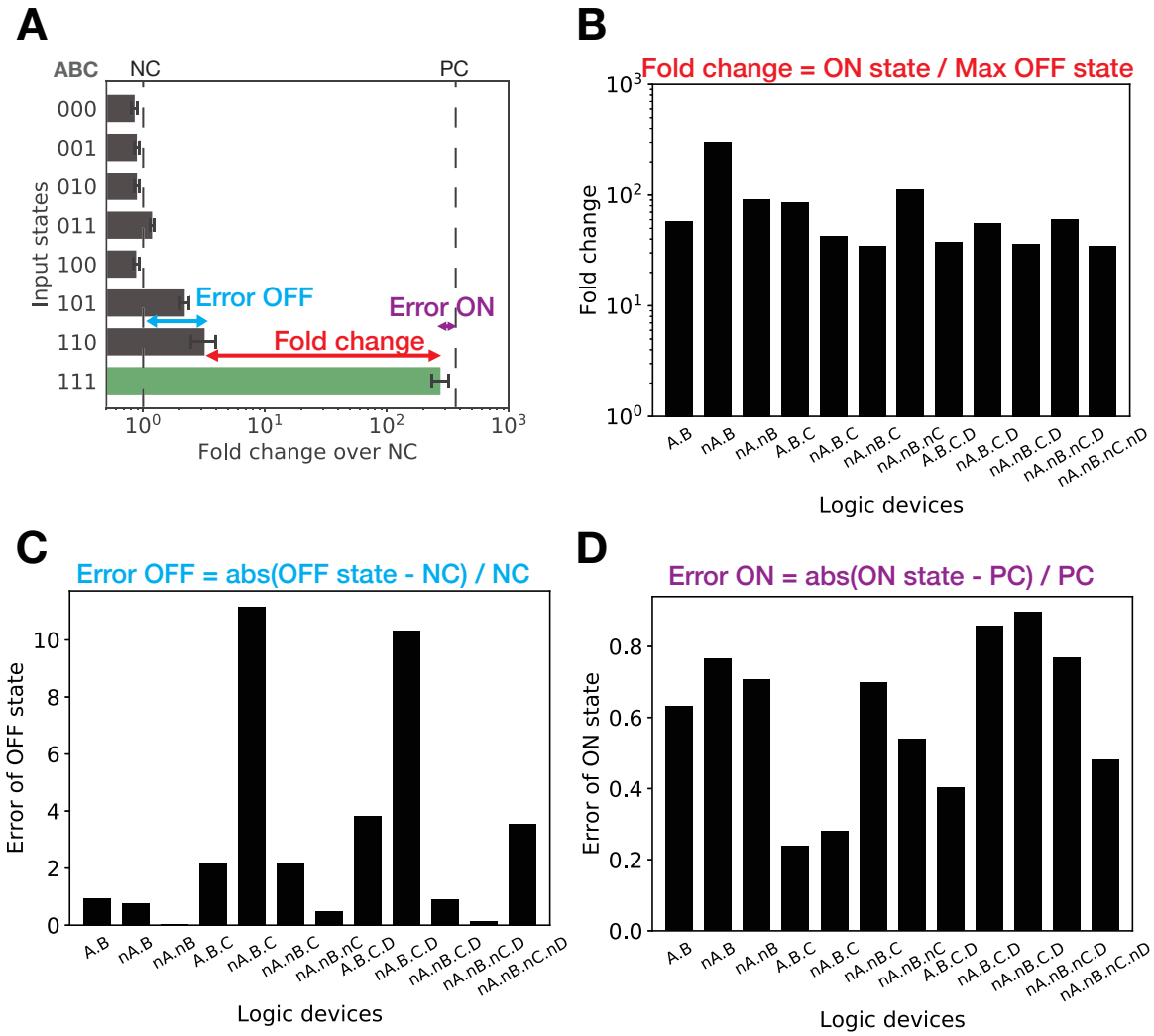


Figure 2.10: **Parameters characterizing our logic devices.** (A) Graphical representation of each parameter, such as the Error OFF between the highest OFF state and the negative control, the Error ON between the ON state and the positive control, and the fold change (the lowest fold change) between the highest OFF state and the ON state. We aim at increasing the fold change and reducing the OFF and ON errors. (B) Fold change for each device represented in a bar graph. The fold change is calculated by dividing the fold change over the negative control (represented in Figure 2.8 and 2.9) of the ON state by the one of the highest OFF state. It corresponds to the minimum fold change between states of each devices. (C) OFF Error for each device represented in a bar graph. The error is calculated as the subtraction of the highest OFF state with the negative control and divided by the negative control. For a perfect device, the error is 0. (D) ON error calculated similarly than for the OFF error, the perfect device would have an ON error of 0.

To address this issue, we thus decided to prototype multicellular logic system operation, asking two fundamental questions: 1) For every state of the truth table, can we discriminate at the population level the expected ON and OFF states? and 2) Can an autonomous multicellular system composed of different strains with varying ON and OFF states actually be sufficiently stable and useable over time?

2.2.5 Prototyping a multicellular system simulating the implementation of complex Boolean logic functions

Our objective here was to simulate the implementation of complex Boolean logic functions which necessitate a multicellular system. As a reminder, for the implementation in living organisms, Boolean logic functions are decomposed into sub-functions. Each sub-function is implemented by one logic device. Strains containing a computational device and the corresponding integrase device are composed to form a multicellular system implementing the complete Boolean function.

In a complete logic circuit, multiple cells should grow together, respond to input signal, switch their computational devices accordingly and produce fluorescence according to the truth table. We did not have all integrases responding to various inputs, we thus simulated the behavior of the final multicellular system by using constitutive integrase cassettes.

To do so, as for the characterization of individual logic devices, each device was co-transformed with constitutive integrase cassettes corresponding to each input state of the truth table. Then, for each input state, strains corresponding to the implementation of each sub-function were mixed in equal proportions. The multicellular cultures corresponding to each input state were grown overnight and the population was characterized via plate reader, flow cytometer and by direct observation on a UV table (Figure 2.11).

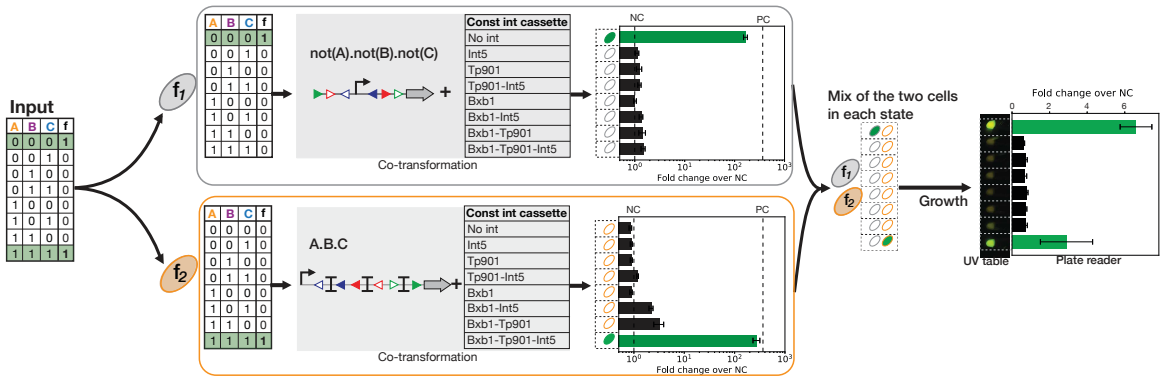


Figure 2.11: **Multicellular system prototyping.** Input truth table is decomposed in sub-truth table corresponding to sub-functions, each implemented in a different strain. Each sub-function is implemented using a specific logic device. Logic devices are co-transformed with constitutive integrase cassettes and the resulting fluorescence intensity of individual strains is measured in the different states. Strains in identical states but containing different logic devices are mixed in equal proportions, and the resulting co-cultures are grown overnight. Bulk fluorescence intensity of each co-culture is analyzed by plate reader and UV table.

2.2.5.1 Detection of GFP output from a multicellular system

We started by prototyping two multicellular systems encoding two different Boolean functions: the 3-input "Consensus" function that necessitates the co-culture of two strains, and a 4-input Boolean function that necessitates the co-culture of three strains, each containing a 3-input logic device (f=A.B.D+not(A).not(B).C+not(A).not(C).not(D)). For the 4-input Boolean function, integrases and inputs are theoretically connected in various manner in each strain therefore, each logic device is co-transformed with the eight constitutive integrase cassettes and the resulting strains are mixed to simulate the 16 4-input states corresponding to the specific connection of inputs with integrases.

We started by observing the fluorescence intensity of co-culture after overnight growth and concentration by centrifugation under a simple UV light table. We clearly observed green fluorescence in the expected ON states and no fluorescence in the OFF state (Figure 2.12A). These data therefore demonstrate the feasibility of our multicellular implementation and the possibility to detect the system output using low-cost equipment.

We also measured the fluorescence intensity of the multicellular system using a plate reader (Figure 2.12A). Here again, we were able to clearly distinguish ON and OFF states. However, we observed a high variability between replicates and an important variability of fluorescence intensity between ON states. As fluorescence intensity of the ON state of each device is different, we expected to observe a difference in the various ON state of the multicellular system. The ON state of A.B.C logic device is higher than the one of not(A).not(B).not(C) logic device; however for the simulated Consensus function we observed the opposite results, as the input state where not(A).not(B).not(C) is ON (000) is higher than the input state where A.B.C is ON (111). These differences are due to difference in proportion of the strains as explained in the next paragraph.

2.2.5.2 Growth competition between strains within a multicellular system

We used flow-cytometry to measure the relative proportions of each subpopulation. Obviously, this quantification could only be performed in a few states, where the different strains had different fluorescent intensity profiles that could be distinguished. We found that this difference in intensity between the replicates was in fact caused by a difference in proportion of cells. As we quantified the relative proportion of sub-populations before growth, we validated that the desired proportions of 50% for the two-strain system and 30% for the three-strain system were obtained using our protocol (see Material and Methods). However, after an overnight growth at 37°C, we observed that the proportion of cells expressing GFP for the 3-input AND gate decreased and was highly variable between replicates (Figure 2.12B-C). In this corresponding input state, the cell containing the A.B.C device express three integrases and GFP. The same effect was observed in the two and three strain systems. These data suggest that constitutive

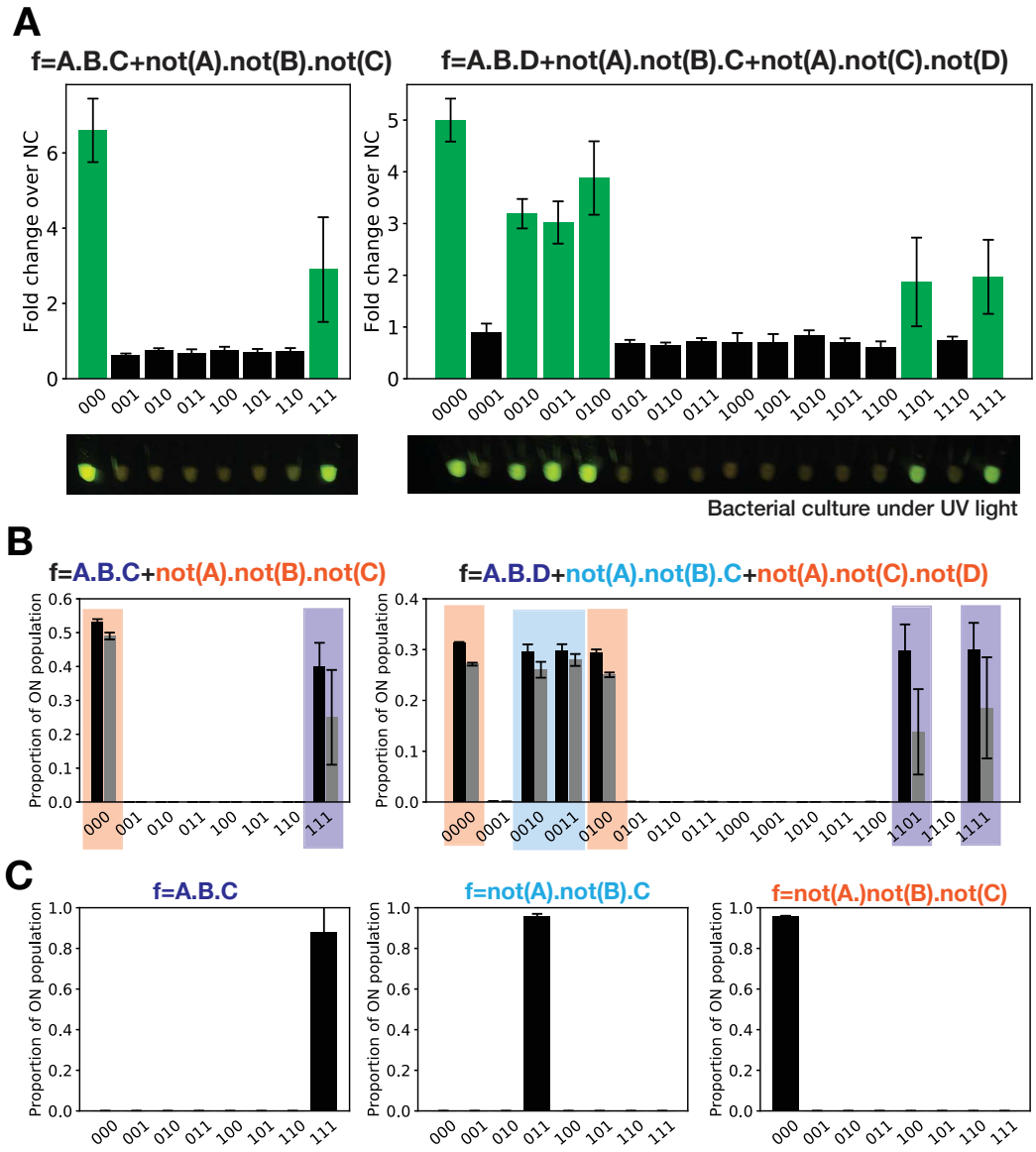


Figure 2.12: **Simulation of multicellular systems for 2 logic functions.** (A) Fluorescence intensity of each co-culture corresponding to the truth table input state of the Boolean function. Bar graphs correspond to plate reader fluorescence intensity measurements of three replicates of co-culture after overnight growth. Fold change over NC (negative control) is the ratio of the fluorescence normalized by the absorbance of the co-culture versus the negative control (more details in Material and Methods). Green bars are for the theoretical ON state of the Boolean function. Pictures corresponded to three replicates of co-cultures of each state centrifuged together, and resuspended in 20 μL and observed under a UV light. (B) and (C) Proportion of ON cells in the total co-culture for each cell. (B) Before (in black) and after overnight growth (in grey), single-cell fluorescence measurements of the co-culture are performed using a flow cytometer. Then, the proportion of cells expressing GFP is defined and plotted in these two graphs. The x-axis corresponds to the input state simulated by the analysed co-culture. (C) The proportion of ON population of the individual strains of each device with each constitutive integrase cassette is represented in these three bar graphs. For (B) and (C), the proportions represented are the mean of three replicates in one experiment and error bars are the error between these three replicates.

expression of these four or five proteins induces a decrease in growth rate and therefore a decrease of the proportion of this population. However, the load on the cellular mechanism should be lower in the final system as the integrase expression would be induced by input signal and not constitutive.

2.2.5.3 Determining a common detection threshold across the computational device family

We wanted to determine if the common fold change between our 14 logic devices was sufficient for multicellular computation. We thus tested a multicellular logic system composed of the logic devices with the highest Error OFF and the logic device with the lowest Error ON value (Figure 2.13). In this worse case scenario, we still clearly discriminated the ON state from the OFF state after overnight growth with a 2.5-fold change. We measured a 10-fold difference between the two ON states. Because the device having the highest ON value is also the one having the highest leakage, we will change the promoter of this logic device to a weaker promoter. By doing so, we should reduce the leakage and obtain a better dynamic range once the multicellular system is assembled.

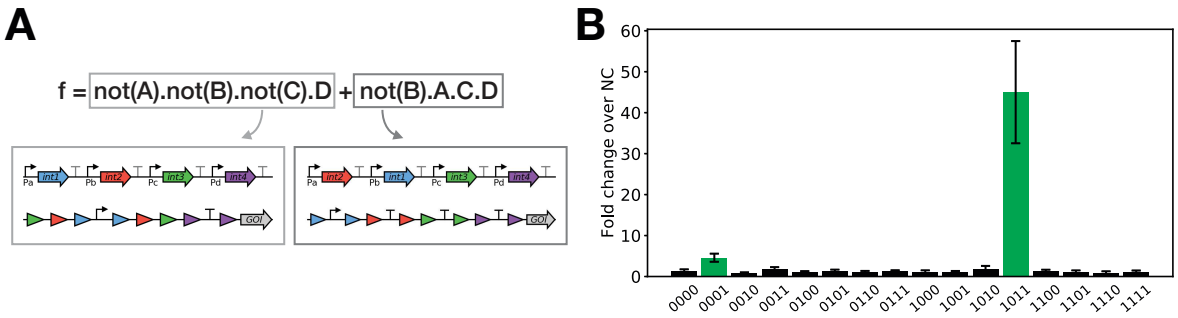


Figure 2.13: **Prototyping the worse case scenario of multicellular logic.** (A) Decomposition of the Boolean function in two strains composed of the logic devices, not(A).not(B).not(C).D and not(A).B.C.D, and different input-integrase connections. (B) Fluorescence intensity of each co-culture simulating this Boolean function in each input state. Data correspond to bulk plate reader measurements of three replicates in one experiment and fold changes over the negative control are obtained as detailed in Material and Methods. Error bars correspond to the error between the three replicates. The green bars correspond to the theoretical ON state of the Boolean function.

Taken together, these data demonstrate the feasibility of composing strains containing recombinase devices to implement complex Boolean functions at the multicellular level. Despite background levels and variability of ON states across our logic devices, we obtained clear and detectable ON states at the multicellular level. Regarding strain competition, enzyme expression could be reduced, and cells could also be compartmentalized in hydrogel beads.

2.2.6 Characterization of parts to optimize logic devices

While characterizing ID- and NOT-elements, we observed important differences of output gene expression for different integrase site orientations. Therefore, we individually characterized each integrase site and a set of 10 terminators in a genetic context identical to our logic devices. We were able to uncover and quantify unknown promoter or terminator activities which were very specific of particular sites and enzymes. This data will be useful for further optimization of logic devices and more generally to any application involving integrase sites.

For characterization of integrase sites, we characterized four sites in the forward and reverse orientation in two different conditions. First, to test their potential effects on transcriptional elongation, we placed integrase sites between the P7 promoter and the RBS gene of interest. To assess the promoter activities of the sites, we placed integrase sites in front of GFP preceded by an RBS. We then characterized these 64 constructs as we previously did for computational elements (Figure 2.14).

For the integrase site-gene cassettes, we expected to have no GFP expression as we did not have any promoter. This was indeed the case for Int5 and Bxb1 sites. However, as observed previously in the characterization of elements, Tp901 and Int7 integrase sites mediate transcription initiation (Figure 2.14A). For Tp901 integrase, the attB site in the forward orientation, the attP site in the reverse orientation, and the attL site in both orientations exhibited a GFP expression of 3- to 10-fold above the negative control. As the attL integrase site is composed of the left arm of the attB site and the right arm of attP site, we can hypothesize that the left part of attB site in forward orientation and the right part of attP site in reverse orientation have promoter activity. According to the Tp901 phage genome, the attP site does not correspond to a promoter region ([Brøndsted 2001]; however, the attB site in the *Lactococcus lactis* genome could be used as a low efficiency promoter. Similarly, for Int7 integrase sites, the attP and attR sites in forward orientation and to lesser extend the attL site in the reverse orientation exhibited GFP expression 2- to 3-fold above the negative control. As both the attP and attR sites in the forward orientation have a low promoter activity and not the attB and attL sites in the forward orientation, we hypothesize that it is the left arm of the attP site of Int7 which initiates low transcription level.

We then tested the termination efficiency of the att sites. Here, if an integrase site did not have any terminator activity, we expected expression of GFP comparable to the positive control. We obtained an important diversity between integrase sites and integrase site orientations for each integrase, and therefore terminator activities for some integrase site orientations (Figure 2.14B). However, the terminator activities measured here are reduced in comparison to promoter activities of some integrase sites. The maximum terminator activity observed was a 2.5-fold reduction of GFP expression (0.4-fold change over PC, or 60% termination efficiency). The terminator activity of integrase sites was clearly unidirectional for Bxb1 integrase. In fact, all integrase sites in the forward orientation induced a 2.5-fold reduction of GFP expression in

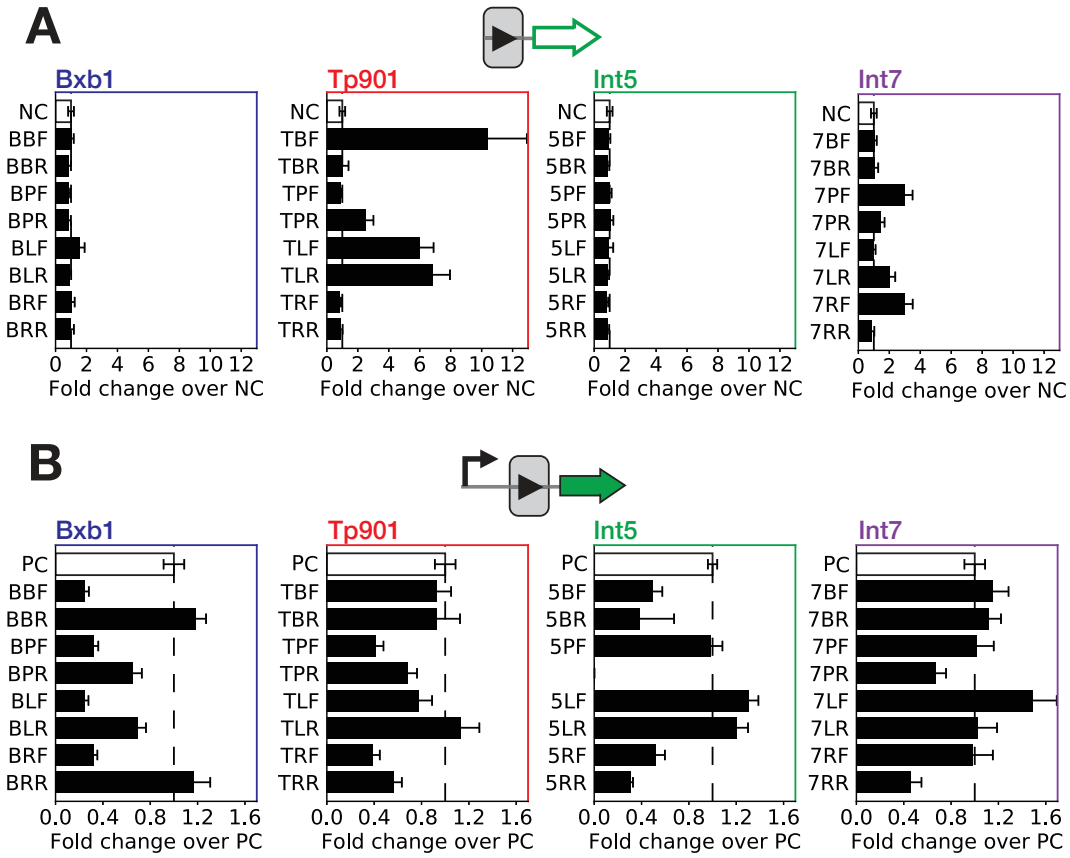


Figure 2.14: **Characterization of integrase sites.** (A) Characterization of promoter activities of integrase sites, each integrase site is positioned in 5,Äô of the gene expression cassette. To named construct, 3 lettres codes are used, with the first letter corresponding to the integrase (B for Bxb1, T for Tp901, 5 for Int5 and 7 for Int7), the second letter corresponding to the type of site (B for attB, P for attP, L for attL and R for attR) and the last letter to the orientation of the site (F for forward and R for reverse). (B) Characterization of the terminator activities of integrase sites, each integrase site is positioned between P7 promoter and the gene expression cassettes. For (A) and (B), The bar graphs correspond to the fold change of mean of fluorescence intensity of constructs over negative control (construct without promoter) for (A) and over positive control (construct with promoter only) for (B). Data were obtained from 3 experiments with 3 replicates per experiments of flow-cytometry measurement. The dash lines correspond to fold change of the negative control (NC, equal to 1) for (A) and the fold change of the positive control (construct with only promoter P7) for (B). Error bars are mean of the standard deviation between the three replicates in each experiments.

comparison to the positive control and from 0- to 1.4-fold decrease for integrase sites in reverse orientation. Nevertheless, for Int5, measured terminator activities are clearly site dependent and not orientation dependent, as attB integrase sites and attR integrase sites induce a 2-fold decrease in GFP expression.

With this characterization, we obtained precise measurements of promoter and terminator activities of each integrase site. It is now clear that integrase sites can have a strong effect on gene expression and it is critical to optimize the orientation of integrase sites in logic devices.

Moreover, based on these data, we might be able to predict logic element behavior using a phenomenological model.

Additionally, we characterized a set of 10 terminators (including the ones selected for the logic elements plus others from [Chen 2013]). As for integrase sites, we characterized terminators with two constructions. In the first construct, the terminator was placed in front of the gene cassette without promoter, and in the second construct, the terminator was placed between the promoter and the output gene (Figure 2.15A).

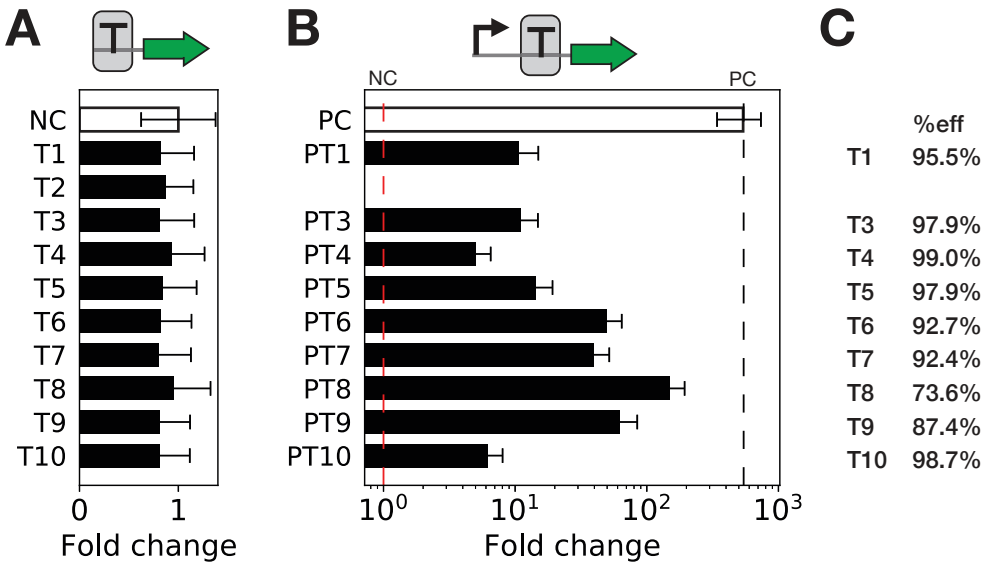


Figure 2.15: **Characterization of terminators.** (A) Characterization of promoter activities of terminators, where each terminator is positioned in 5' of the gene expression cassette. (B) Characterization of the terminator activities of each terminator, where each terminator is positioned between P7 promoter and the gene expression cassettes. For (A) and (B), bar graphs correspond to the fold change in mean fluorescence intensity over negative control (construct without promoter). Data were obtained from three experiments with three replicates per experiment of flow-cytometry measurement. Error bars are the mean of the standard deviation between the three replicates in each experiment. For (B), the dashed lines correspond to fold change over the negative control (NC, equal to 1) and the fold change over the positive control (PC, construct with only promoter P7). (C) Terminator efficiency of each terminator calculated from the previous characterization in (B). Terminator efficency is calculated as one minus the difference between positive control GFP fluorescence and terminator GFP fluorescence and divided by positive control. The formula is detailed in Materials and Methods.

For the first constructs, we did not detect GFP expression; therefore, terminators do not have promoter activities, which is comforting. The second set of constructs permits us to quantify the activity of each terminator in the conditions of our logic devices. Indeed, terminators have previously been characterized by placing terminators between two output genes and measuring the ratio of expression of the two genes. In our case, the terminator is expected to block the RNA polymerase directly after initiation and therefore in a different genetic context. We indeed obtained terminator efficiencies different from the ones previously characterized (Figure

2.15B). We obtained transcription efficiency (corresponding to the difference in GFP expression without terminator and with terminator divided by the GFP expression without terminator) of between 73.6% to 99% (Figure 2.15C). We did not manage to clone the construct with promoter-T2, probably due to the important secondary structure of the construct. Except for T2, terminators selected for our logic devices show an efficiency of termination between 95% to 99%. To optimize computational elements, we could switch the Bxb1 terminator to T10 to increase termination efficiency from 95 to 98%. Otherwise, the selected terminators have the best efficiency of transcription termination of the characterized set.

2.2.7 Discussion

In this work, we engineered logic devices by hierarchical composition of well-characterized and optimised logic elements. We characterized NOT- and ID-elements for four different integrases in various integrase site orientations. This characterization allowed us to select optimal logic elements. We composed these selected elements to obtain the 14 logic devices required for implementing all 4-input Boolean logic functions. Without further optimization, these 14 logic devices behaved as expected, with ON and OFF states corresponding to the Boolean function that they implement. This set of devices shows a better fold change than previously designed biological logic gates and for the first time permits implementation of all 4-input logic functions. Nevertheless, the logic devices designed by composition of selected elements are not perfect; we obtained a high variability between ON states of logic devices and high background gene expression in OFF states. Despite this, all simulated multicellular systems demonstrated the feasibility of the composition of these logic devices to implement complex Boolean functions in multicellular system. Moreover, in our simulated multicellular systems, the output detection was possible using a simple and low cost UV light.

To calibrate all ON and OFF states of logic devices, we will change the promoter of devices with high OFF background level and high ON level to a weaker promoter (e.g. the P6 promoter), such as for the not(A).B.C.D and not(A).B.C devices. This should permit an increase in the common fold change of devices, as the highest OFF level will be lower, and furthermore it will create a uniform ON level. This thresholding of gates is essential for multicellular systems but also for the design of multi-layered integrase-based circuits.

This strategy of design of a complex system by decomposition into parts at different levels, such as decomposition into strains, into elements, and into parts simplifies the optimization and debugging of large circuits.

By characterizating the set of NOT- and ID-elements, we realized the important effect of integrase sites on gene expression. Therefore, we characterized the promoter and terminator activities of all integrase sites in all orientations. We observed important promoter activities of Tp901 sites and terminator activities of Bxb1 sites into forward orientation. Sites which are identified to have a strong influence on gene expression could be re-engineered to decrease this

effect. One possibility would be to mutate the right arm of the attP Tp901 integrase sites, which have been identified as the most probable cause of promoter activity. More generally, these data would be useful for the design of any integrase-based system, such as logic gates, but also as landing pads for site-specific integration.

As we characterized individual parts, elements, and logic devices, we started to generate a simple phenomenological model that permits from these data to predict element and logic device behaviors. This model could allow us to further optimize logic device designs. Other logic device designs based on the same elements are possible by alternating the position of ID-elements and of integrase sites surrounding a promoter for NOT-elements. Moreover, in this work, we tried to optimize logic device behavior, but this led to devices with variable output ON levels. Using a predictive model, we could predict the behavior of all possible devices and select a set with the best common threshold and fold change and not specially the best ones.

Our simulation of a multicellular system illustrates the feasibility of these designs. Various proportions of ON cells were obtained after overnight growth likely due to the important load to the cell. To address this concern, the experiment might require further optimization such as growth at different temperatures. To avoid issues with differences in growth rates between cells, one option would be to encapsulate cells in alginate beads. Beads would then be placed in the same environment to detect the input signals. As no cell-to-cell communication is required in our multicellular system, encapsulation would simplify the use of this system for various applications. Moreover, the output state can also be detected by sequencing. Indeed, as using integrase, the state of the system would be encoded in the DNA in an irreversible manner. Therefore, by sequencing, the output state can be determined even if the cells are dead or if the ON signal is not detectable due to a decrease of proportion of the ON population. Finally, to obtain a constant output level, cell-to-cell communication via quorum sensing could be used to integrate the output signal (see design in Annex D). However, implementation of this cell-to-cell communication will require important design optimization.

To permit the implementation of a complete multicellular logic system, we need to connect integrases to inducible promoters in a streamlined manner. Our logic devices are independent of input signal, but they require connection of all inputs to all integrases. Therefore, if we want to be able to implement all Boolean logic functions for all applications (all input signals), we have to be able to engineer robust integrase switches responding to any input signal. This is not straightforward as due to the irreversibility of the integrase switches and his high catalytic activity, a small leakage in integrase gene expression induces irreversible switches of the system. One strategy to connect inducible promoters to integrases is to tune the translation and protein degradation efficiency. We tried to connect the inducible promoter PyeaR with the Int5 integrase using various ribosome binding sites and degradation tags, but most of the constructs were still leaky. More efforts are now underway for the development of a workflow to systematically generate inducible switches. One option is to decrease the plasmid copy number

of the integrase generator. This work will be a key step in permiting the use of our logic devices for implementation of Boolean logic function in living organisms.

After final optimization, we will distribute our logic devices in open access platform, such as Addgene, to permit implementation of all Boolean logic functions. Our devices are designed and characterized in *E. coli* but a similar workflow is applicable to any living organisms as integrases function in various organisms such as mouse, zebrafish, *S. cerevisiae*, and *Drosophila*.

We believe that this design workflow by decomposition into simple parts and the distribution of our logic devices will help researchers and engineers to reprogram cellular behavior for various applications in a streamlined manner.

2.2.8 Materiel and Methods

2.2.8.1 Workflow: High-throughput design and cloning of DNA constructs

Due to the large set of constructs that we have to build, it was essential to define an standard cloning workflow amenable to parallelization and eventually high-throughput.

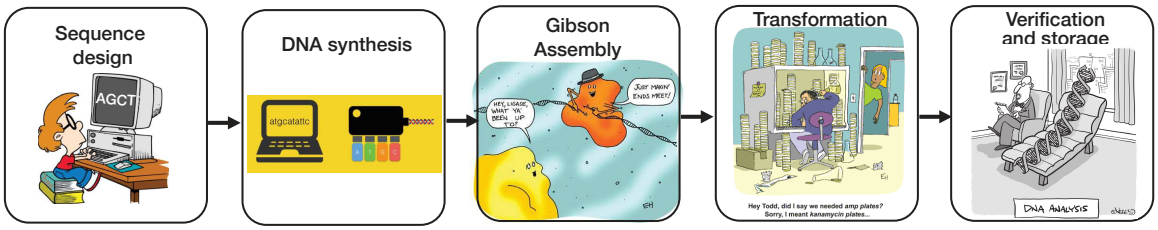


Figure 2.16: Explanation bellow

1 - For sequence design, a large number of software prgorams exists such as Gene designer; however, they are designed to handle a reduced number of sequences. Therefore, we wrote a python script to automatically generate our library of DNA sequences. The use of a script minimizes the number of errors. Moreover, because the final sequences result from permutations of a reduced set of parts, Python is particularly well suited for the task. All sequences were designed to support cloning by Gibson assembly at an identical location in our template vector, such as between spacer 0 and the beginning of the GFP sequence. Consequently, all sequences were composed of the 40 bp spacer 0 in the 5' end, a variable sequence corresponding to the logic element, and RiboJ, BCD2 and 40 bp of the beginning of the GFP sequence.

2 - All sequences were synthesized as linear fragment.

3 - Before receiving the linear fragments, **the vector is prepared for Gibson assembly**. PCR is performed with a primer reverse of sp0 and a forward primer for the beginning of the GFP to obtain linearized vector ready to be assembled with each linear insert fragment.

4 - When the linear fragments are received, they are resuspended and directly mixed with the linearized vector and the Gibson assembly mix to perform a **Gibson assembly reaction**. Afterward one hour of incubation, Gibson assembly reactions are transformed into chemically competent *E. coli* DH5alphaZI.

5 - High-throughput chemical transformation. Upstream of the transformation, a large batch of homemade chemical competent cells were prepared and aliquoted in PCR strip tubes. Therefore, the chemical transformation is performed mainly using multichannel pipettes and could therefore be easily adapt to a pipetting robot. After heat shock, cells are transferred to 96 well plates previously filled with SOC. For plating cells, 6 well-plates are used, which simplifies manipulations and reduces the quantity of media and material needed. The 6-well plates are filled with LB agar, completed with the appropriate antibiotic, and well-dried before use.

The 96-well plate containing the transformed cells is spin at 4,000 rpm for 5 min to concentrate the cells in a smaller volume by removing the majority of the supernatant. The remaining volume of each well is plated in one well of a 6-well agar plate. Spreading bacteria on the plate is performed with glass beads.

6 - Colony PCR is performed to verify the insertion of a fragment of the correct size in the vector. The protocol of colony PCR is detailed in Material and Methods. Two colonies are tested per Gibson assembly; one of the two is selected for further plasmid extraction and DNA sequencing.

7 - Plasmid extraction is performed on one colony per construct. Colonies are grown overnight in 24 deep well plates with 5 mL of LB. Standard plasmid extraction protocol is used afterward. Constructs are verified by DNA sequencing.

Therefore, from the design to the obtaining of the constructs a total of 14 working days are required, with 10 days for the synthesize and 4 for the cloning and DNA sequencing. A second round of plasmid extraction and DNA sequencing can be required if the first results are not conclusive. For the construction of the logic elements, 80% of the constructs were successfully obtained after the first round.

2.2.8.2 *E. coli* strains and media

The DH5alphaZ1 *E. coli* strain was used in this study (laciq, PN25-tetR, SpR, deoR, supE44, Delta(lacZYA-argFV169), Phi80 lacZDeltaM15, hsdR17(rK- mK+), recA1, endA1, gyrA96, thi-1, relA1). *E. coli* were grown on LB media with antibiotic corresponding to the transformed plasmid(s). Antibiotics were purchased from Sigma and used at the following concentration: chloramphenicol 20μg/mL, kanamycin 25μg/mL, carbenicillin 50μg/mL (for ampicillin resistance). For co-transformation of two plasmids, the two corresponding antibiotics were used at the previously defined concentration divided by two.

2.2.8.3 Molecular biology

We used pSB4K5 and J66100 (from parts.igem.org) as vectors. The pSB4K5 plasmid is composed of a Kanamycin resistance cassette and pSC101 low copy origin of replication and was used for the cloning of BP and LR targets, parts, elements, and devices. J66100 plasmid is composed of ampicillin resistance cassette and ColE1 origin of replication, and was used for the cloning of integrase cassettes. All plasmids used in this study were derived from these two vectors and fragments were assembled using one-step isothermal assembly following standard molecular biology procedures. Enzymes for the one-step isothermal assembly were purchased from New England BioLabs (NEB, Ipswich, MA, USA). PCR were performed using Q5 PCR master mix and One-Taq quick load master mix for colony PCR (NEB). Primers were purchased

from IDT (Louvain, Belgium) and DNA fragments from Twist Bioscience (San Francisco, CA, USA). Plasmid extraction and DNA purification were performed using kits from Biosentec (Toulouse, France). Sequencing was cinducted by GATC Biotech (Cologne, Germany).

2.2.8.4 Construction of BP and LR targets

For Tp901 and Bxb1 targets, the BP and LR targets from Bonnet et al. were used. For Int3, Int4, Int5, and Int7 targets, a template sequence composed of mKate in reverse orientation and GFP in reverse was synthesized and assembled in pSB4K5 with sp0 and spN as homology region and using P862 and P863 to linearized pSB4K5 vector. Then, target fragments containing the sequence between the mKate and GFP coding sequences were synthesized and assembled using P109 and P224 to linearized the previously constructed template sequence.

2.2.8.5 Construction of parts, elements, and devices

As a backbone sequence, the expression operating unit from Guiziou et al. (*B. subtilis* toolbox) was used, which composed of the spacers for Gibson assembly, superfold GFP, and terminator. The construct was inserted in pSB4K5 using P862 and P863. For the construction of NOT- and IDENTITY-elements, and positive and negative constructs, the previous construct was used as a template and amplified using P71 and P870 for one-step isothermal assembly with linear fragments corresponding to each element. For the integrase sites, terminators, and logic devices, the terminator in 3' side of the construct was switched from B0015 to L3S3P00. As B0015 was used in the logic devices, we wanted to avoid sequence homology in the construct. The linear DNA fragment sp5_ L3S3P00_ spN was inserted between sp5 and spN spacers using P40 and P34 for vector amplification to switch the terminator in the positive control construct. Therefore, for integrase sites, terminators, and logic devices, the positive control construct with L3S3P00 as terminator was used for insertion of DNA fragment with P71 and P870 for vector amplification.

Short name	Original name	Used for	DNA sequence
T1	ECK120033737	Bxb1 ID-element in excision	ggaaacacagAAAAAAGCCCGCACCTGACAGTGCGGGCTTTTTTTTTcgaccaaagg
T2	ECK120029600	Tp901 ID-element in excision	TTCAGCCAAAAAACTTAAGACCGCCGGTCTTGTCCACTACCTTGCAGTAATGCGGTGGACAGGATCGGCGGTTTTCTTTTCTCTTCTCAA
T3	L3S2P21	Int7 ID-element in excision	CTCGGTACCAAATTCCAGAAAAGAGGCCTCCCGAAAGGGGGGCCTTTTTTCGTTTTGGTCC
T4	L3S3P21	Int5 ID-element in excision	CCAATTATTGAAGGCCTCCCTAACGGGGGGCCTTTTTTTGTTTCTGGTCTCCC
T5	B0015	Bxb1 ID-element in inversion	ccaggcatcaaataaaacgaaaggctcagtcgaaagactgggcctttcgttttatctgttgtttgtcggtgaacgctctctactagagtcacactggctcaccttcgggtgggcctttctgcgtttata
T6	J61048	Tp901 ID-element in inversion	ccggcttatcggtcagtttcacctgatttacgtaaaaacccgcttcggcgggtttttgcttttggagggggcagaaagatgaatgactgtccacgacgctatacccaaaagaaa
T7	ECK120015170	Int7 ID-element in inversion	ACAATTTTCGAAAAAACCCGCTTCGGCGGGTTTTTTTATAGCTAAAA
T8	ECK120010855	Int5 ID-element in inversion	GTAACAACGGAAACCGGCCATTGCGCCGGTTTTTTTTGGCCT

Table 2.2: List of terminator used in the different elements.

2.2.8.6 Construction of integrase cassettes

First, Plac-Integrase cassettes were synthetized and cloned in the J66100 plasmid. The landing pad construct with promoters, terminators, and spacers and without integrase genes was synthesized and assembled in J66100 using P1122 and P1153 to linearized the vector. Each integrase was amplified from the previous Plac construct and inserted separately in the landing pad using the following primers:

Short name	Integrase expressed	Vector Primers	Insert Primers
C-Int1	Bxb1	P1325-P1326	P1324-P1327
C-Int2	Tp901	P1329-P1330	P1328-P1331
C-Int3	Int5	P1333-P1334	P1332-P1335
C-Int4	Int7	P1337-P1338	P1336-P1339

Table 2.3: Construction of single-integrase constitutive cassettes.

For construction of all the integrase cassette variants, the integrase cassettes with a single integrase were used in combinatorial manner. The following is a table with the corresponding assembly process.

Short name	Integrases present	Insert 1	Insert 2	Insert 3	Insert 4
C-Int5	Bxb1-Tp901	Bxb1-sp23R/sp19F	Tp901-sp23F/sp19R		
C-Int6	Bxb1-Int5	Bxb1-sp23R/sp19F	Int5-sp23F+SP19R		
C-Int7	Bxb1-Int7	Bxb1-sp23R/sp19F	Int7-sp23F/sp19R		
C-Int8	Tp901-Int5	Tp901-sp25R/sp19F	Int5-sp25F/sp19R		
C-Int9	Tp901-Int7	Tp901-sp25R/sp19F	Int7-sp25F/sp19R		
C-Int10	Int5-Int7	Int5-sp27R/sp19F	Int7-sp27F/sp19R		
C-Int11	Bxb1-Tp901-Int5	Bxb1-sp23R/sp19F	Tp901-sp23F/sp25R	Int5-sp25F/sp19R	
C-Int12	Bxb1-Tp901-Int7	Bxb1-sp23R/sp19F	Tp901-sp23F/sp25R	Int7-sp25F/sp19R	
C-Int13	Bxb1-Int5-Int7	Bxb1-sp23R/sp19F	Int5-sp23F/sp27R	Int7-sp27F/sp19R	
C-Int14	Tp901-Int5-Int7	Tp901-sp25R/sp19F	Int5-sp25F/sp27R	Int7-sp27F/sp19R	
C-Int15	Bxb1-Tp901-Int5-Int7	Bxb1-sp23R/sp19F	Tp901-sp23F/sp25R	Int5-sp25F/sp27R	Int7-sp27F/sp19R

Table 2.4: Assembly of constitutive integrase cassettes

2.2.8.7 Flow-cytometer measurements

Quantification of expression levels of all strains was performed using an Attune NxT flow cytometer (ThermoFisher) equipped with an autosampler. Experiments were performed in 96 well plates with three replicates per plate. For flow cytometry measurements, 20, 000 bacterial events were analysed. A gate was previously designed based on forward and side scatter graphs to remove debris from the analysis. GFP fluorescence intensity was measured using excitation by a 488 nm laser and a 510/10 nm filter (BL1). RFP excitation was performed by a 561 nm laser and a 615/25 nm filter (YL2). Voltages used were FFS: 440, SSC: 340, BL1: 360, for all experiments except with BP and LR targets, and BL1: 400 and YL2: 400, for experiments with BP and LR targets. Data were analysed and presented using the Flow-Jo (Tristar) software.

2.2.8.8 Characterization of integrases: cell culture, measurement, and analysis

For integrase characterization, each Plac-integrase plasmid and dual controller for Tp901 integrase was co-transformed with BP targets. For constitutive integrase cassette characterization, each constitutive integrase cassette was co-transformed with the BP targets corresponding to the integrase that it should express.

For both experiments, 96 deep-well plates filled with 500 μL of LB per wells were inoculated with three clones per co-transformation and three clones per control corresponding to the BP target and LR target strains. For integrase characterization with Plac-integrase plasmid and dual controller plasmid, LB was supplemented with 100 μM of IPTG for co-transformation with Plac-integrase and 1% arabinose for co-transformation with the dual controller for expression of Tp901. Plates were grown for 16 hours at 37°C. Cultures were diluted 40 times with Focusing Fluid and directly measured on the flow cytometer according to previously described methods.

Data analysis was performed using Flow-Jo. Bacterial events were gated to remove debris from the analysis by plotting FSC-H over SSC-H. Data were represented using a density plot of BL1-H over YL2-H, corresponding to the GFP fluorescence intensity over the RFP fluorescence intensity. For the Figure 2.1D and Figure 2.6E, the proportions of bacteria in BP or LR states were obtained using a BL1-H over YL2-H plot by gating the population corresponding to the BP or LR target strain. Data represented in the heatmap correspond to the mean of the proportion obtained for the three replicates in one experiment.

2.2.8.9 Characterization of elements, integrase sites and terminators

Glycerol stocks from each construct were streaked on plates. 96 deep-well plates filled with 500 μL of LB and kanamycin antibiotic were inoculated with three clones from the freshly streaked plates. For all experiments, three clones of the negative control strain corresponding to GFP without promoter and the positive control strain corresponding to P7-GFP were inoculated.

Plates were grown for 16 hours at 37°C. Cultures were diluted 40 times with Focusing Fluid and measured using a flow cytometer. Three experiments with three replicates per experiments were performed for elements, integrase sites, and terminators characterizations. Data were analyzed using Flow-Jo. Bacterial events were gated to remove debris from the analysis by plotting FSC-H over SSC-H. For each independent experiment, the median GFP fluorescence intensity of the bacterial population for each replicate was extracted, corresponding to the BL1-H median. Then, the mean fluorescence intensity and the standard deviation between replicates were calculated from the three replicates in a single experiment. The mean values from three or two independent experiments were then calculated. The error was calculated as the mean of the standard deviation for each experiment, corresponding therefore to the mean error between replicates in one experiment. The fold change over the negative control represented in the bar graph was obtained by dividing the mean fluorescence intensity value from several independent experiments of the construct of interest by the mean fluorescence intensity value of the negative control. The error on the fold change was calculated from the mean error between replicates of the construct of interest and the negative control.

$$Error\ on\ fold\ change = \left(\frac{Error_{construct}}{Fluo_{construct}} + \frac{Error_{negative\ control}}{Fluo_{negative\ control}}\right) \times \frac{Fluo_{construct}}{Fluo_{negative\ control}}$$

2.2.8.10 Characterization of devices

Each device was characterized with different integrase cassettes corresponding to the number of inputs for the device. To do so in a streamlined way, chemical competent cells of *E. coli* strains with each constitutive integrase cassette were prepared and aliquoted in PCR strips, with an 8-tube PCR strip for 3-input constitutive integrase cassettes and an 8-tube PCR strip for additional constitutive integrase cassettes required for 4-input characterization. Detailed protocol for chemical competent cells and transformation can be found in Annex. Devices were transformed in competent cells corresponding to the number of inputs. Transformations were plated on 6-well plates filled with 3 mL of LB agar to reduced the quantity of plates required. For 2-input devices, 4 transformations were performed, 8 for 3-input devices and 16 for 4-input devices, leading to a total of 124 transformations. For each transformation, three clones were picked and inoculated in 500 μL of LB in 96 deep-well plates. Additionally, the negative control (GFP without promoter) strain and positive control (P7-GFP) strain were streaked from glycerol stocks and three clones were picked and inoculated. Plates were grown for 16 hours at 37°C. Cultures were diluted 40 times with Focusing Fluid and measured on a flow cytometer. Two experiments with three replicates per experiments were performed. Data were analyzed using Flow-Jo using the same procedure as the one detailed previously for element, integrase site, and terminator characterizations.

2.2.8.11 Simulation of multicellular logic system

To simulate multicellular logic systems, required devices were transformed in competent cells with the corresponding constitutive integrase cassettes, as for device characterisation. Three clones per transformation were inoculated in 500 μL of LB in 96 deep-well plates. Plates were incubated for 16 hours at 37°C to reach stationary phase. From the stationary phase culture, cells were mixed in identical proportions: such as 100 μL for each culture in a 96 deep-well plate to simulate the multicellular logic system (see following tables for more detail). To quantify the proportion of ON cells in each state before growth, these mixes of cell cultures were diluted 200 times in Focusing Fluid and measured with a flow cytometer.

For growth, each mix of cultures- was diluted 1000 times in LB in two serial dilutions: 10 μL in 190 μL of LB and 10 μL in 500 μL of LB in the final 96 deep-well plate. Plates were incubated for 16 hours at 37°C. Co-cultures were diluted four times in PBS and analyzed using a plate reader for measurement of bulk fluorescence intensity. Additionally, co-cultures were diluted 200 times in Focusing Fluid and analyzed on a flow cytometer. Finally, the three replicates were mixed, centrifuged, and cell pellets were resuspended in 20 μL in PCR tubes and imaged under a UV table.

Plate reader measurements were performed using a BioTek Cytation 3. GFP fluorescence intensity (Excitation: 485 nm, Emission: 528 nm, gain: 85) and absorbance at 600 nm were measured. For each sample, GFP fluorescence intensity over absorbance at 600nm were calculated and the mean value was calculated between the three replicates. The fold change over the negative control was determined from this mean value over the one of the negative control. The error bars correspond to the error of the fold change: the sum of the coefficient of variation between replicates of the construct and of the negative control multiplied by the calculated fold change. Flow-cytometry experiments were performed as detailed in the corresponding section. To determine the proportion of cells in the ON state (expressing GFP), a first gate was performed to select bacterial events using the FSC-H over SSC-H density plot. A second gate was performed from bacterial events to select single cells using SSC-A over SSC-H density plot. Finally, from single cell events, a BL1-H histogram was plotted and cells with more than 4, 200 fluorescence intensity in arbitrary units were considered ON to determine the proportion of ON cells using a final gate. This procedure was used to analyse flow-cytometry experiments before and after co-culture growth.

Simulated input state	BACD - 40	ABCD- 42
0000	C-Int0	C-Int0
1000	C-Int2	C-Int1
0100	C-Int1	C-Int2
0010	C-Int3	C-Int3
1100	C-Int5	C-Int5
1010	C-Int8	C-Int6
0110	C-Int6	C-Int8
1110	C-Int11	C-Int11
0001	C-Int4	C-Int4
1001	C-Int9	C-Int7
0101	C-Int7	C-Int9
0011	C-Int10	C-Int10
1101	C-Int12	C-Int12
1011	C-Int14	C-Int13
0111	C-Int13	C-Int14
1111	C-Int15	C-Int15

Table 2.5: Simulation of A.B.C+not(A).not(B).not(C) logic function implementation - Figure 2.12

Simulated input state	(ABD) 35	(ABC) 37	(ACD) 38
0000	C-Int0	C-Int0	C-Int0
0001	C-Int3	C-Int0	C-Int3
0010	C-Int0	C-Int3	C-Int2
0011	C-Int3	C-Int3	C-Int8
0100	C-Int2	C-Int2	C-Int0
0101	C-Int8	C-Int2	C-Int3
0110	C-Int2	C-Int8	C-Int2
0111	C-Int8	C-Int8	C-Int8
1000	C-Int1	C-Int1	C-Int1
1001	C-Int6	C-Int1	C-Int6
1010	C-Int1	C-Int6	C-Int5
1011	C-Int6	C-Int6	C-Int11
1100	C-Int5	C-Int5	C-Int1
1101	C-Int11	C-Int5	C-Int6
1110	C-Int5	C-Int11	C-Int5
1111	C-Int11	C-Int11	C-Int11

Table 2.6: Simulation of A.B.D+not(A).not(B).C+not(A).not(C).not(D) logic function implementation - Figure 2.12

Simulated input state	BACD - 40	ABCD- 42
0000	C-Int0	C-Int0
1000	C-Int2	C-Int1
0100	C-Int1	C-Int2
0010	C-Int3	C-Int3
1100	C-Int4	C-Int4
1010	C-Int6	C-Int5
0110	C-Int5	C-Int6
1110	C-Int7	C-Int7
0001	C-Int8	C-Int8
1001	C-Int10	C-Int9
0101	C-Int9	C-Int10
0011	C-Int11	C-Int11
1101	C-Int12	C-Int12
1011	C-Int14	C-Int13
0111	C-Int13	C-Int14
1111	C-Int15	C-Int15

Table 2.7: Simulation of not(A).not(B).not(C).D+not(B).A.C.D logic function implementation - Figure 2.13

2.2.8.12 List of Constructs

Short Name	Pasmid	Description	Used for
BP target Bxb1	pSB4K5	Prefix-RFP-attB-Pfwd-attP-GFP-Suffix	integrase characterisation
LR target Bxb1	pSB4K5	Prefix-RFP-attL-Prev-attR-GFP-Suffix	integrase characterisation
BP target Tp901	pSB4K5	Prefix-RFP-attB-Pfwd-attP-GFP-Suffix	integrase characterisation
LR target Tp901	pSB4K5	Prefix-RFP-attL-Prev-attR-GFP-Suffix	integrase characterisation
BP target Int3	pSB4K5	sp0-mKate-attB-Prev-attP-sfGFP-spN	integrase characterisation
LR target Int3	pSB4K5	sp0-mKate-attL-Pfwd-attR-sfGFP-spN	integrase characterisation
BP target Int4	pSB4K5	sp0-mKate-attB-Prev-attP-sfGFP-spN	integrase characterisation
LR target Int4	pSB4K5	sp0-mKate-attL-Pfwd-attR-sfGFP-spN	integrase characterisation
BP target Int5	pSB4K5	sp0-mKate-attB-Prev-attP-sfGFP-spN	integrase characterisation
LR target Int5	pSB4K5	sp0-mKate-attL-Pfwd-attR-sfGFP-spN	integrase characterisation
BP target Int7	pSB4K5	sp0-mKate-attB-Prev-attP-sfGFP-spN	integrase characterisation
LR target Int7	pSB4K5	sp0-mKate-attL-Pfwd-attR-sfGFP-spN	integrase characterisation
Plac.Bxb1	J66100	sp0-Plac-Bxb1-spN	integrase characterisation
Plac.Int3	J66100	sp0-Plac-Int3-spN	integrase characterisation
Plac.Int4	J66100	sp0-Plac-Int4-spN	integrase characterisation
Plac.Int5	J66100	sp0-Plac-Int5-spN	integrase characterisation
Plac.Int7	J66100	sp0-Plac-Int7-spN	integrase characterisation
Dual controler	J64100		integrase characterisation
C-Int0	J66100	Landing pad for constitutive integrase sp18sp19	device characterisation
C-Int1	J66100	Landing pad with Bxb1	device characterisation
C-Int2	J66100	Landing pad with Tp901	device characterisation
C-Int3	J66100	Landing pad with Int5	device characterisation
C-Int4	J66100	Landing pad with Int7	device characterisation
C-Int5	J66100	Landing pad with Bxb1-TP901	device characterisation
C-Int6	J66100	Landing pad with Bxb1-Int5	device characterisation
C-Int7	J66100	Landing pad with Bxb1-Int7	device characterisation
C-Int8	J66100	Landing pad with Tp901-Int5	device characterisation
C-Int9	J66100	Landing pad with Tp901-Int7	device characterisation
C-Int10	J66100	Landing pad with Int5-Int7	device characterisation
C-Int11	J66100	Landing pad with Bxb1-Tp901-Int5	device characterisation
C-Int12	J66100	Landing pad with Bxb1-Tp901-Int7	device characterisation
C-Int13	J66100	Landing pad with Bxb1-Int5-Int7	device characterisation
C-Int14	J66100	Landing pad with Tp901-Int5-Int7	device characterisation
C-Int15	J66100	Landing pad with Bxb1-Tp901-Int5-Int7	device characterisation
CM-N	pSB4K5	sp0-GFP-B0015-spN	Negative control
CM-P	pSB4K5	sp0-P7-GFP-B0015-spN	Positive control
CM-P L3	pSB4K5	sp0-P7-GFP-L3S3P00-spN	Positive control

Table 2.8: List of constructs

Short Name	Pasmid	Description	Used for
B.1	pSB4K5	P-BF-PF-1	element
B.2	pSB4K5	P-BR-PR-1	element
B.3	pSB4K5	P-PF-BF-1	element
B.4	pSB4K5	P-PR-BR-1	element
B.5	pSB4K5	Pinv-BF-PR-1	element
B.6	pSB4K5	Pinv-BR-PF-1	element
B.7	pSB4K5	Pinv-PF-BR-1	element
B.8	pSB4K5	Pinv-PR-BF-1	element
B.9	pSB4K5	T-BF-PF-1	element
B.10	pSB4K5	T-BR-PR-1	element
B.11	pSB4K5	T-PF-BF-1	element
B.12	pSB4K5	T-PR-BR-1	element
B.13	pSB4K5	Tinv-BF-PR-1	element
B.14	pSB4K5	Tinv-BR-PF-1	element
B.15	pSB4K5	Tinv-PF-BR-1	element
B.16	pSB4K5	Tinv-PR-BF-1	element
T.1	pSB4K5	P-BF-PF-2	element
T.2	pSB4K5	P-BR-PR-2	element
T.3	pSB4K5	P-PF-BF-2	element
T.4	pSB4K5	P-PR-BR-2	element
T.5	pSB4K5	Pinv-BF-PR-2	element
T.6	pSB4K5	Pinv-BR-PF-2	element
T.7	pSB4K5	Pinv-PF-BR-2	element
T.8	pSB4K5	Pinv-PR-BF-2	element
T.9	pSB4K5	T-BF-PF-2	element
T.10	pSB4K5	T-BR-PR-2	element
T.11	pSB4K5	T-PF-BF-2	element
T.12	pSB4K5	T-PR-BR-2	element
T.13	pSB4K5	Tinv-BF-PR-2	element
T.14	pSB4K5	Tinv-BR-PF-2	element
T.15	pSB4K5	Tinv-PF-BR-2	element
T.16	pSB4K5	Tinv-PR-BF-2	element
5.1	pSB4K5	P-BF-PF-3	element
5.2	pSB4K5	P-BR-PR-3	element
5.3	pSB4K5	P-PF-BF-3	element
5.4	pSB4K5	P-PR-BR-3	element
5.5	pSB4K5	Pinv-BF-PR-3	element
5.6	pSB4K5	Pinv-BR-PF-3	element
5.7	pSB4K5	Pinv-PF-BR-3	element

Table 2.8: List of constructs

Short Name	Pasmid	Description	Used for
5.8	pSB4K5	Pinv-PR-BF-3	element
5.9	pSB4K5	T-BF-PF-3	element
5.10	pSB4K5	T-BR-PR-3	element
5.11	pSB4K5	T-PF-BF-3	element
5.12	pSB4K5	T-PR-BR-3	element
5.13	pSB4K5	Tinv-BF-PR-3	element
5.14	pSB4K5	Tinv-BR-PF-3	element
5.15	pSB4K5	Tinv-PF-BR-3	element
5.16	pSB4K5	Tinv-PR-BF-3	element
7.1	pSB4K5	P-BF-PF-4	element
7.2	pSB4K5	P-BR-PR-4	element
7.3	pSB4K5	P-PF-BF-4	element
7.4	pSB4K5	P-PR-BR-4	element
7.5	pSB4K5	Pinv-BF-PR-4	element
7.6	pSB4K5	Pinv-BR-PF-4	element
7.7	pSB4K5	Pinv-PF-BR-4	element
7.8	pSB4K5	Pinv-PR-BF-4	element
7.9	pSB4K5	T-BF-PF-4	element
7.10	pSB4K5	T-BR-PR-4	element
7.11	pSB4K5	T-PF-BF-4	element
7.12	pSB4K5	T-PR-BR-4	element
7.13	pSB4K5	Tinv-BF-PR-4	element
7.14	pSB4K5	Tinv-BR-PF-4	element
7.15	pSB4K5	Tinv-PF-BR-4	element
7.16	pSB4K5	Tinv-PR-BF-4	element
31	pSB4K5	Tp901 ID-element with B0015	element
32	pSB4K5	A.B	devices
33	pSB4K5	not(A).B	devices
34	pSB4K5	not(A).not(B)	devices
35	pSB4K5	A.B.C	devices
36	pSB4K5	not(A).B.C	devices
37	pSB4K5	not(A).not(B).C	devices
38	pSB4K5	not(A).not(B).not(C)	devices
39	pSB4K5	A.B.C.D	devices
40	pSB4K5	not(A).B.C.D	devices
41	pSB4K5	not(A).not(B).C.D	devices
42	pSB4K5	not(A).not(B).not(C).D	devices
43	pSB4K5	not(A).not(B).not(C).not(D)	devices
BBF	pSB4K5	attB Foward Bxb1 - GFP	integrase site characterisation
BBR	pSB4K5	attB Reverse Bxb1 - GFP	integrase site characterisation
BPF	pSB4K5	attP Foward Bxb1 - GFP	integrase site characterisation
BPR	pSB4K5	attP Reverse Bxb1 - GFP	integrase site characterisation

Table 2.8: List of constructs

Short Name	Pasmid	Description	Used for
BLF	pSB4K5	attL Foward Bxb1 - GFP	integrase site characterisation
BLR	pSB4K5	attL Reverse Bxb1 - GFP	integrase site characterisation
BRF	pSB4K5	attR Foward Bxb1 - GFP	integrase site characterisation
BRR	pSB4K5	attR Reverse Bxb1 - GFP	integrase site characterisation
TBF	pSB4K5	attB Foward Tp901 - GFP	integrase site characterisation
TBR	pSB4K5	attB Reverse Tp901 - GFP	integrase site characterisation
TPF	pSB4K5	attP Foward Tp901 - GFP	integrase site characterisation
TPR	pSB4K5	attP Reverse Tp901 - GFP	integrase site characterisation
TLF	pSB4K5	attL Foward Tp901 - GFP	integrase site characterisation
TLR	pSB4K5	attL Reverse Tp901 - GFP	integrase site characterisation
TRF	pSB4K5	attR Foward Tp901 - GFP	integrase site characterisation
TRR	pSB4K5	attR Reverse Tp901 - GFP	integrase site characterisation
5BF	pSB4K5	attB Foward Int5 - GFP	integrase site characterisation
5BR	pSB4K5	attB Reverse Int5 - GFP	integrase site characterisation
5PF	pSB4K5	attP Foward Int5 - GFP	integrase site characterisation
5PR	pSB4K5	attP Reverse Int5 - GFP	integrase site characterisation
5LF	pSB4K5	attL Foward Int5 - GFP	integrase site characterisation
5LR	pSB4K5	attL Reverse Int5 - GFP	integrase site characterisation
5RF	pSB4K5	attR Foward Int5 - GFP	integrase site characterisation
5RR	pSB4K5	attR Reverse Int5 - GFP	integrase site characterisation
7BF	pSB4K5	attB Foward Int7 - GFP	integrase site characterisation
7BR	pSB4K5	attB Reverse Int7 - GFP	integrase site characterisation
7PF	pSB4K5	attP Foward Int7 - GFP	integrase site characterisation
7PR	pSB4K5	attP Reverse Int7 - GFP	integrase site characterisation
7LF	pSB4K5	attL Foward Int7 - GFP	integrase site characterisation
7LR	pSB4K5	attL Reverse Int7 - GFP	integrase site characterisation
7RF	pSB4K5	attR Foward Int7 - GFP	integrase site characterisation
7RR	pSB4K5	attR Reverse Int7 - GFP	integrase site characterisation
P-BBF	pSB4K5	P7 - attB Foward Bxb1 - GFP	integrase site characterisation
P-BBR	pSB4K5	P7 - attB Reverse Bxb1 - GFP	integrase site characterisation
P-BPF	pSB4K5	P7 - attP Foward Bxb1 - GFP	integrase site characterisation
P-BPR	pSB4K5	P7 - attP Reverse Bxb1 - GFP	integrase site characterisation
P-BLF	pSB4K5	P7 - attL Foward Bxb1 - GFP	integrase site characterisation
P-BLR	pSB4K5	P7 - attL Reverse Bxb1 - GFP	integrase site characterisation
P-BRF	pSB4K5	P7 - attR Foward Bxb1 - GFP	integrase site characterisation
P-BRR	pSB4K5	P7 - attR Reverse Bxb1 - GFP	integrase site characterisation
P-TBF	pSB4K5	P7 - attB Foward Tp901 - GFP	integrase site characterisation
P-TBR	pSB4K5	P7 - attB Reverse Tp901 - GFP	integrase site characterisation
P-TPF	pSB4K5	P7 - attP Foward Tp901 - GFP	integrase site characterisation
P-TPR	pSB4K5	P7 - attP Reverse Tp901 - GFP	integrase site characterisation

Table 2.8: List of constructs

Short Name	Pasmid	Description	Used for
P-TLF	pSB4K5	P7 - attL Foward Tp901 - GFP	integrase site characterisation
P-TLR	pSB4K5	P7 - attL Reverse Tp901 - GFP	integrase site characterisation
P-TRF	pSB4K5	P7 - attR Foward Tp901 - GFP	integrase site characterisation
P-TRR	pSB4K5	P7 - attR Reverse Tp901 - GFP	integrase site characterisation
P-5BF	pSB4K5	P7 - attB Foward Int5 - GFP	integrase site characterisation
P-5BR	pSB4K5	P7 - attB Reverse Int5 - GFP	integrase site characterisation
P-5PF	pSB4K5	P7 - attP Foward Int5 - GFP	integrase site characterisation
P-5PR	pSB4K5	P7 - attP Reverse Int5 - GFP	integrase site characterisation
P-5LF	pSB4K5	P7 - attL Foward Int5 - GFP	integrase site characterisation
P-5LR	pSB4K5	P7 - attL Reverse Int5 - GFP	integrase site characterisation
P-5RF	pSB4K5	P7 - attR Foward Int5 - GFP	integrase site characterisation
P-5RR	pSB4K5	P7 - attR Reverse Int5 - GFP	integrase site characterisation
P-7BF	pSB4K5	P7 - attB Foward Int7 - GFP	integrase site characterisation
P-7BR	pSB4K5	P7 - attB Reverse Int7 - GFP	integrase site characterisation
P-7PF	pSB4K5	P7 - attP Foward Int7 - GFP	integrase site characterisation
P-7PR	pSB4K5	P7 - attP Reverse Int7 - GFP	integrase site characterisation
P-7LF	pSB4K5	P7 - attL Foward Int7 - GFP	integrase site characterisation
P-7LR	pSB4K5	P7 - attL Reverse Int7 - GFP	integrase site characterisation
P-7RF	pSB4K5	P7 - attR Foward Int7 - GFP	integrase site characterisation
P-7RR	pSB4K5	attR Reverse Int7 - GFP	integrase site characterisation
T1	pSB4K5	ECK120033737 - GFP	terminator characterisation
T2	pSB4K5	ECK120029600 - GFP	terminator characterisation
T3	pSB4K5	L3S2P21 - GFP	terminator characterisation
T4	pSB4K5	L3S3P21 - GFP	terminator characterisation
T5	pSB4K5	B0015 - GFP	terminator characterisation
T6	pSB4K5	J61048 - GFP	terminator characterisation
T7	pSB4K5	ECK120015170 - GFP	terminator characterisation
T8	pSB4K5	ECK120010855 - GFP	terminator characterisation
T9	pSB4K5	L3S2P11 - GFP	terminator characterisation
T10	pSB4K5	L3S3P22 - GFP	terminator characterisation
P-T1	pSB4K5	P7 - ECK120033737 - GFP	terminator characterisation
P-T2	pSB4K5	P7 - ECK120029600 - GFP	terminator characterisation
P-T3	pSB4K5	P7 - L3S2P21 - GFP	terminator characterisation
P-T4	pSB4K5	P7 - L3S3P21 - GFP	terminator characterisation
P-T5	pSB4K5	P7 - B0015 - GFP	terminator characterisation
P-T6	pSB4K5	P7 - J61048 - GFP	terminator characterisation
P-T7	pSB4K5	P7 - ECK120015170 - GFP	terminator characterisation
P-T8	pSB4K5	P7 - ECK120010855 - GFP	terminator characterisation
P-T9	pSB4K5	P7 - L3S2P11 - GFP	terminator characterisation
P-T10	pSB4K5	P7 - L3S3P22 - GFP	terminator characterisation

Table 2.8: List of constructs

Chapter 3

Programming history-dependent logic in a multicellular system

Contents

In this chapter, I will present my work on the implementation of history-dependent logic in a multicellular system. This work is presented as a paper composed of two parts: (1) the development of an automated design framework and (2) the experimental implementation of this design.

I have done this work in collaboration with Jerome Bonnet, Ana Zuniga, and Pauline Mayonove. Jerome Bonnet and myself were at the origin of the project. The Python software and web-interface was created simultaneously for Boolean logic and history-dependent logic implementation. As with chapter 2, I was at the origin of the design workflow and its automation via

the creation of the Python software. Violaine Moreau and Laurent Bonnet were involved in the creation of the CALIN website. The implementation and characterization of history-dependent gene-expression programs were performed by Ana Zuniga, Pauline Mayonove, and myself. Ana Zuniga performed the multi-cellular experiments and Pauline Mayonove the characterization of the 3-input OSiRIS constructs.

3.1 Introduction

Survival and reproduction of living organisms depends on their highly sophisticated abilities to sense, process, and respond to multiple signals in parallel [Bray 1995]. While several biological circuits operate in real-time, responding to particular signals according to past events is also crucial. Such history-dependent biological responses are observed from animal behavior down to the heart of fundamental processes like cellular differentiation and morphogenesis [Wolpert 2015]. Microorganisms may also be capable of some forms of history-dependent behavior that could confer a fitness advantage during the evolutionary competition [Wolf 2008].

From a research and engineering perspective, the ability to generate synthetic history-dependent genetic programs has many practical implications. First, such programs would allow the study and understanding of complex biological phenomena in which time dependencies are important (e.g. development). Second, history-dependent programs would enable the implementation of sophisticated behaviors not found in nature (e.g. biological counters), thereby pushing the frontiers of biological systems engineering [Collins 2017].

Recently, researchers have started exploring the implementation of time-dependent biological programs. In these history-dependent programs, gene expression outcome depends on the order of occurrence of signals. In order to encode history-dependent behavior, molecular memory devices capable of recording past events are needed, and different designs have been used to do so.

The genetic toggle switch [Toman 1985, Gardner 2000] was used as the basis for building a Push-on/Push-off switch [Lou 2010] or to engineer a circuit producing a response comparable to Pavlovian behavior in *E. coli* [Zhang 2014]. In another example, a cascade "counter-like" device uses RNA molecules to store the occurrence of inputs, producing an output after a certain number of stimuli separated by user-defined lag times [Friedland 2009].

But among all systems, recombinase memory devices quickly emerged as the tool of choice to implement history-dependent behavior. Recombinase memory devices are based on the inversion or excision of DNA sequences via site-specific recombinases [Podhajska 1985, Ham 2006]. Recombinases, in particular serine integrases, have been used to encode complex Boolean logic devices within living cells using reduced, single-layer architectures [Bonnet 2013, Siuti 2013, Weinberg 2017]. Contrary to feedback-based systems, recombinase devices exhibit a dual nature in which the state of the system can be encoded both into its gene expression state as well as into the DNA sequence. Every state transition corresponds to a discrete physical change in the DNA sequence.

Because recombinase switches exhibit memory, recombinase logic gates are asynchronous devices that can respond to multiple signals even if they arrive independently. For example, a 2-input recombinase AND gate will produce an output in response to one signal only if the other signal has already been present in the past. However, because recombination reactions

are independent in Boolean devices, the end-point state of the system is the same regardless of the order of occurrence of the signals.

On the other hand, by interlacing target sites of different recombinases, recombination reactions can be made dependent on one another. The system can transition through different DNA states depending on which recombination reaction occurs first [Ham 2008b]. Using this concept, researchers started to implement genetic devices tracking the order of signal occurrences, as well as history-dependent gene expression programs [Ham 2008b, Hsiao 2016, Roquet 2016].

Previous works designed a scaffold for tracking all possible combinations of events using interlaced pairs of mutant recombination sites [Roquet 2016]. This scaffold was shown to be sufficient to produce a different DNA state for every possible state of a 3-input sequential tree (16 states). In order to design history-dependent programs, the authors used this scaffold and computationally generated a combinatorial library of all possible gene expression constructs. Several 3-input programs were successfully implemented based on this library. However, it is not clear if all programs are accessible. It might not be the case due to the architectural constraints imposed by the initially chosen scaffold.

In addition, the scalability of such systems might be challenging for several reasons. First, each program is executed using an ad-hoc design, requiring a case-by-case optimization. Second, it is not clear how many pairs of mutant recombination sites can be used in parallel in a single cell without any non-specific recombination reaction occurring [Colloms 2014]. Third, repetitive DNA sequences often lead to genetic instability through homologous recombination [Nielsen 2016, Sleight 2013] and are notoriously difficult to synthesize.

We thus aimed at designing a composition framework enabling all possible history-dependent gene-expression programs for up to five inputs to be systematically implemented within a multicellular system. To this aim, we conceived of modular scaffolds specifically designed to control gene expression. We took advantage of the division of labor within a multicellular system and used distributed multicellular computation as a means to obtain a composable system (Figure 3.1).

Based on distributed multicellular computation, we developed a modular and scalable design framework for history-dependent gene-expression programs. Our design framework does not require brute-force computation, is scalable to five inputs and uses a reduced number of modular scaffold.

We introduce a method to optimize history-dependent programs called OSiRIS (Optimization by Synthesis of Intermediate Recombination States). Because every state of the system corresponds to a particular physical state of the target DNA sequence, we were able to synthesize and characterize each intermediate state separately. It is consequently an important advantage for optimization to have the state of the system encoded within the DNA sequence, a feature absent in feedback-based systems.

In order to make our design framework broadly accessible to the scientific community, we provide a web server for designing up to 5-input history-dependent gene-expression programs.

3.2 Automated design of history-dependent programs

3.2.1 Distributing history-dependent gene-expression programs within a multicellular system

Each history-dependent gene-expression program can be represented as a lineage tree (Figure 3.1A for two inputs). In a lineage tree, each node corresponds to a state of the inputs. The output of each state is represented in the corresponding node by a color or a number (e.g. black for gene expression and white for no expression). Each lineage corresponds to a specific order-of-occurrence of the inputs. The number of lineages is equal to N! where N is the number of inputs. For instance, for 2 inputs, 2 lineages exist, while for 3 inputs, 6 lineages exist. In our design, we decomposed the history-dependent gene-expression program into subprograms corresponding to the different lineages. Each subprogram is then performed by a different strain subpopulation (Figure 3.1B). Of note, we consider that the system operates in fundamental mode, i.e. inputs do not occur simultaneously, but sequentially.

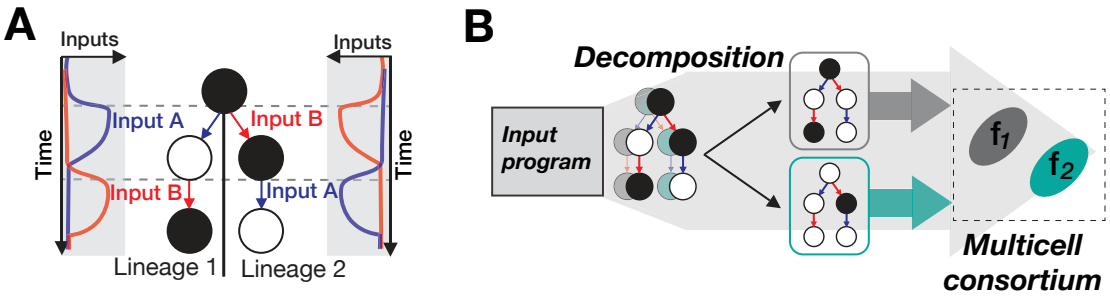


Figure 3.1: **Decomposition of history-dependent gene-expression programs.** (A) Representation of history-dependent gene-expression program in a lineage tree. For a 2-input system (A and B), these can occur in two orders, either A then B or B then A, where these orders correspond to the two lineages of our lineage tree. Arrows represent the occurrence of inputs and nodes the state of the system. The output genes expressed in a specific input states are represented by a specific color or/and number in the node. (B) The program is decomposed into sub-programs, each program corresponding to a different lineage. Each sub-program (f1, f2) is implemented in a different strain. The composition of the strains in a multicellular system permits implementation of the full program.

3.2.2 A modular scaffold design to implement history-dependent gene expression programs

We then designed a modular scaffold capable of executing all possible 2-input history-dependent gene expression programs that can occur within a single lineage. The scaffold contains three

directional cloning positions, each supporting expression of a corresponding gene of interest (GOI) in a particular state of the lineage tree (Figure 3.2A). Thus, any possible combination of gene expression states within a particular lineage can be achieved by simply inserting the desired gene at a given position (Figure 3.2B). Importantly, depending on the identity of the different GOIs, the scaffold can be used to support single or multiple output programs.

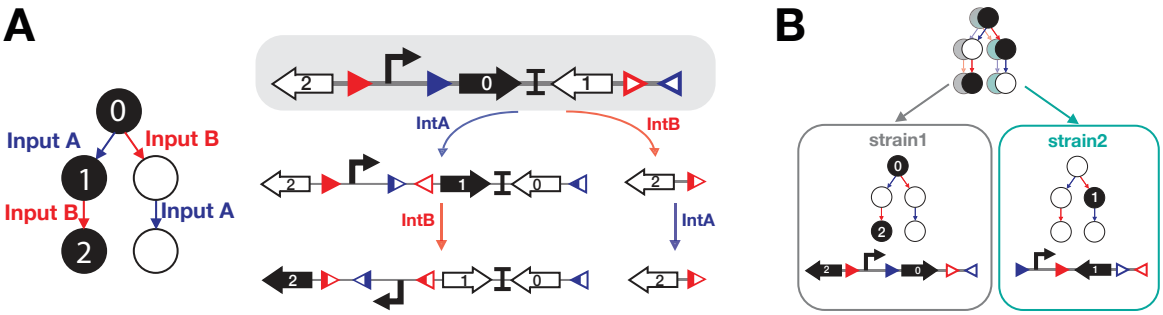


Figure 3.2: **Implementation of 2-input history-dependent gene-expression programs** (A) 2-input history-dependent scaffold. Integrase sites are positioned to permit expression of an output gene in the corresponding lineage. Therefore, for each state of the lineage a different gene is expressed. No gene is expressed for states which are not in the lineage. On the right panel, gene 0 is expressed only when no input is present. If input A is present first, gene 1 is expressed, but if input B is present first, no gene is expressed (nor will be expressed) as the promoter is excised. If input B follows input A, gene 2 is expressed. (B) Example of the implementation of a 2-input history-dependent gene-expression program in a multicellular system. As states are ON in the two different lineages, each lineage is implemented in a different strain using a history-dependent logic device corresponding to the 2-input scaffold with integrase sites at specific positions corresponding to the lineage and output genes at the corresponding GOI positions.

Cellular subpopulations containing a scaffold incorporating different GOIs can be combined to perform a multi-lineage history-dependent genetic output (Figure 3.2B). If control signals are exchanged between the different integrases, the same scaffold can be reused in all lineages.

Scaling up the 2-input scaffold, we designed scaffolds for 3-, 4-, and 5-input history-dependent gene-expression programs (Figure 3.3A-B). The 3- and 4-input scaffolds allow for expression of a different GOI in each state of a given lineage (Figure 3.3A for 3-inputs), while the 5-input scaffold allows expression of a different GOI in each state except in the state 0 (no input). An additional strain is needed if gene expression is required in this input state.

The maximum number of cellular computation units needed to implement a history-dependent gene expression program is equal to the number of lineages (N! for N inputs). A maximum of 6 strains is needed for 3-input programs and 24 strains for 4-input programs (Figure 3.3C). However, most functions are implementable with fewer than the maximum number of cells.

As an example if multiple-output programs, a 3-input/3-output history-dependent gene-expression program represented in Figure 3.3D required two strains. Three different output

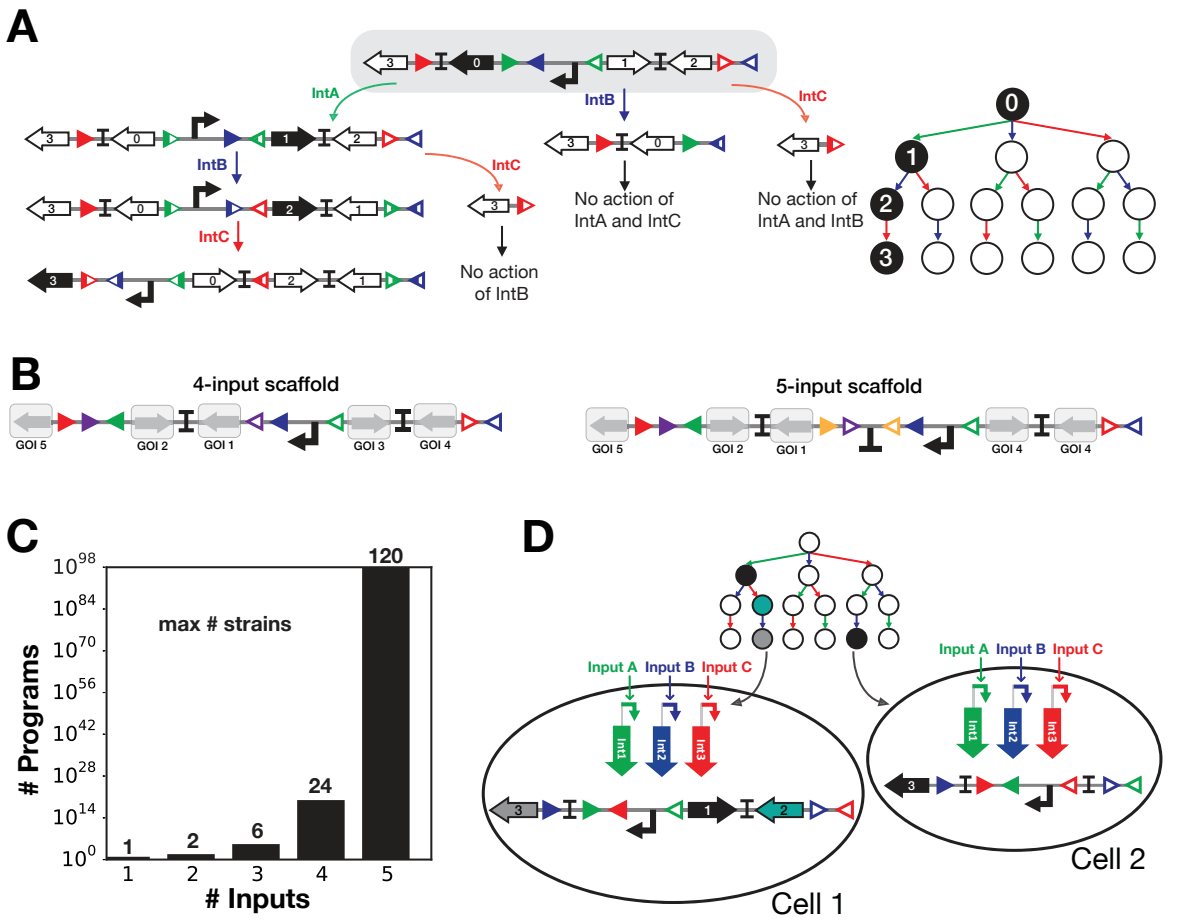


Figure 3.3: **Scaling-up implementation of history-dependent programs to 5 inputs.** (A) 3-input scaffold design for the A then B then C lineage. The scaffold is composed of 4 GOI positions, in each state of the lineage a different output gene is expressed and no gene is expressed in the input states of different lineages. (B) 4- and 5-input scaffold designs. Designs follow the same principles as in A for 3 inputs. For the 5-input scaffold, no gene is expressed when no input is present. (C) Number of 1-output history-dependent programs and maximum number of strains needed for 1 to 5 inputs. The bar graph represents the number of 1-output history-dependent programs from 1 to 5 inputs and the number at the top of each bar corresponds to the maximum number of strains required for implementation of N-input programs. See materials and methods for detailed equations. (D) Example of a 3-input and 3-output history-dependent gene-expression program. The input lineage tree corresponding to the history-dependent program is composed of 4 ON states with 3 different outputs in two lineages. This program is implemented in 2 different strains, one for each lineage. The first cell computes the lineage A then C then B (Green-Red-Blue) with each ON state corresponding to a different output. Consequently, different types of output genes are inserted in the corresponding GOI positions. For the second strain, the lineage implemented is C then A then B (Red-Green-Blue); integrase sites are positioned specifically to implement this lineage and the output gene is positioned in the corresponding position.

genes are placed in the corresponding GOI positions and the three inputs are connected differently to integrases in the two different strains (Figure3.3D). For expression of multiple outputs in a single history-dependent state, the output genes are positioned in a polycistronic architecture at the same GOI position.

In summary, our scaffold-based design supports the execution of up to 5-input/N-output history-dependent gene-expression programs within a multicellular population.

Additionally, we found that a basic history-dependent motif could be repeatedly distributed into different cells to straightforwardly implement all input event-order trackers using a multicellular system (Annex E). The state of the tracker could be addressed experimentally via multiplexed next-generation sequencing.

3.2.3 Automation of history-dependent gene-expression program designs

We encoded an algorithm to automate the design of history-dependent program using Python (Figure 3.4). The algorithm takes a lineage tree as input (equivalent to a sequential truth table). The output corresponds to the biological implementation, such as a graphical representation of the genetic circuit and its associated DNA sequences for each strain.

In our Python algorithm, the lineage tree is decomposed into sub-trees corresponding to ON states in a single lineage (Figure 3.4A). This decomposition is performed by iteratively subtracting the lineages containing ON states. Multiple decompositions are possible. To obtain the fewest number of subprograms, our algorithm prioritizes lineages with ON output-states with the highest number of inputs present (i.e. from the right to the left of the lineage tree). After decomposition, two pieces of information for each selected lineage are extracted: the identity of ON states and the corresponding lineage. Using this information, the history-dependent logic device is constructed. The identity of the integrase sites are determined by the lineage and the position and identity of GOI by the identity of ON states. By combining the logic device design of the different lineages, we obtain the design to implement the input history-dependent gene expression program.

To enable broad access to our design framework, we provide a website for systematic and automated design of history-dependent logic called CALIN (Composable Asynchronous Logic using Integrase Networks). This web-interface is accessible at: `http://synbio.cbs.cnrs.fr/calin/sequential_input.php`. In the CALIN web-interface, the user fills in the number of inputs to process and the desired sequential truth table. The interface provides as an output the DNA architectures of the computational devices and the connection map between signals and integrases along with the corresponding DNA sequences generated for *E. coli*.

In addition to this web interface, the algorithm written in Python is available on Github and can be directly used for high-throughput generation of biological designs.

3.2.4 Minimization of history-dependent circuits using Boolean logic devices

The number of strains required for implementing history-dependent gene-expression programs can be reduced using Boolean logic devices. Indeed, gene-expression programs independent of

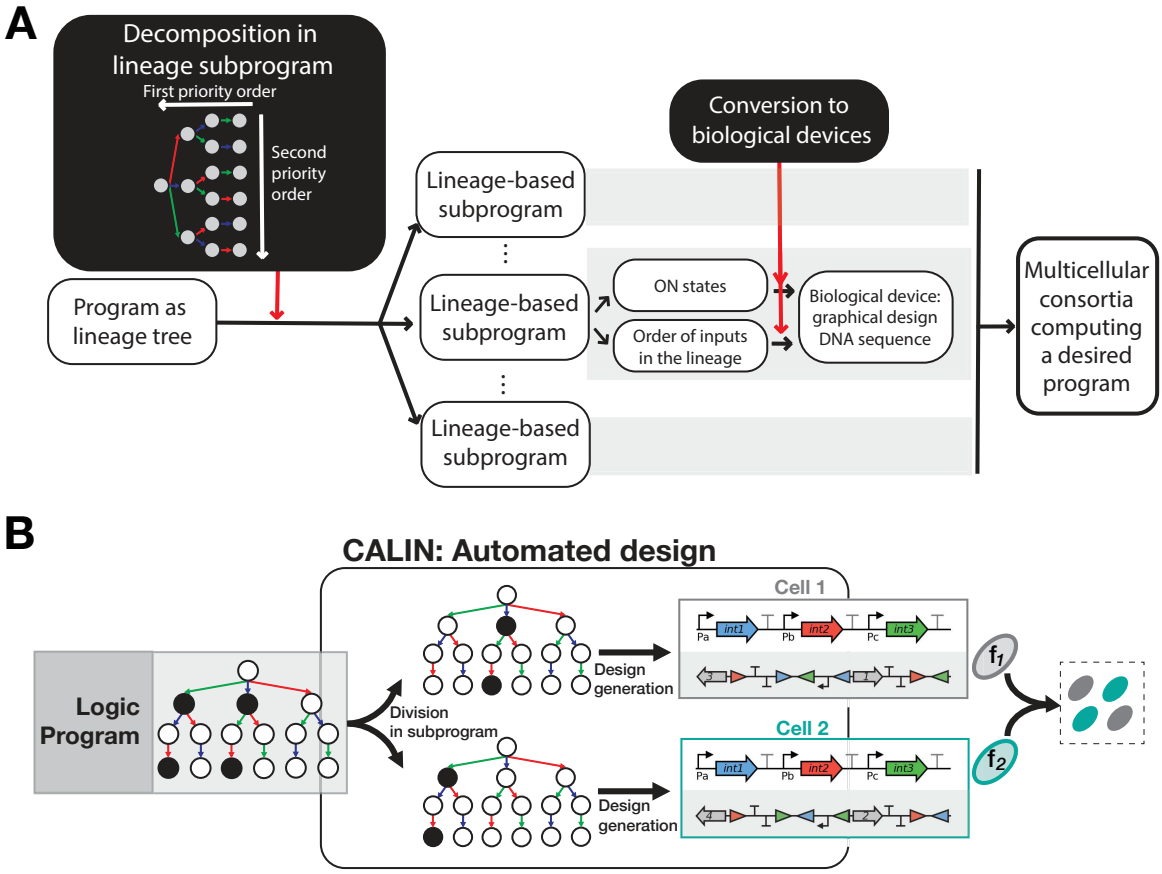


Figure 3.4: **Automated design of history-dependent programs using the CALIN web-interface.** (A) Design algorithm. The Python program takes as input a history-dependent program written as a lineage tree. This program is decomposed into sub-programs, and the decomposition is performed by preferentially extracting subprograms with ON state at the extremity of the tree (corresponding to state with the highest number of inputs present). For each subprogram, the algorithm identifies the identity of ON states and the order of the inputs in the lineage. Based on this information, the biological design is obtained with the graphical design of the integrase cassette and the history-dependent device and the DNA sequence of the device. By composition of the designs of each subprogram in different strains the full program design is obtained. (B) The CALIN web-interface, as following the previously described algorithm, takes as input the logic program as a lineage tree and gives as output the graphical design and DNA sequence of the device for each subprogram.

the history of occurrence of inputs are implementable using Boolean logic devices as detailed in Chapter 2 (Figure 3.5A). Some history-dependent gene-expression programs are decomposable into Boolean logic function(s) and history-dependent subprogram(s). The combination of history-dependent and Boolean logic devices allows a reduction in the number of strains required for the implementation of some history-dependent gene-expression programs. For example, the 3-input history-dependent program represented in Figure 3.5B is decomposable into a 3-input Boolean function and a history-dependent program, by combining Boolean and history-dependent devices, only two strains versus the six strains needed using only history-dependent devices. Additionally, even without reducing the number of strains, the use of Boolean logic

devices instead of history-dependent devices can allow a reduction in the size of the circuit.

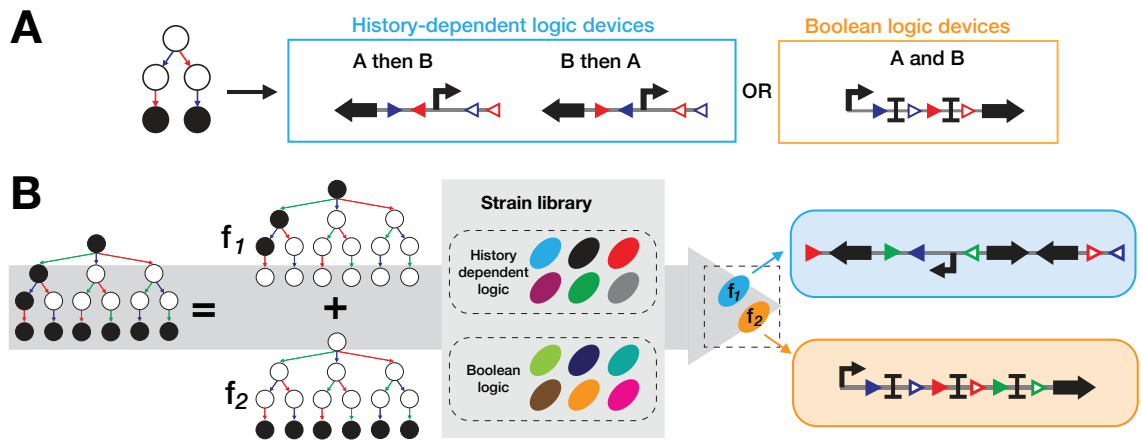


Figure 3.5: **Minimization by simplification of history-dependent programs into Boolean-logic programs.** (A) Functional equivalence between history-dependent logic devices and Boolean logic devices. The 2-input program (left side) can be either implemented using two history-dependent logic devices (A then B and B then A) or with one Boolean logic device (A and B). (B) Example of a 3-input program (initially decomposed in six lineages) that can be simplified into two subprograms using Boolean logic devices. The first subprogram corresponds to a lineage tree with three ON states in a single lineage (one strain). The second subprogram corresponds to a lineage tree with six ON states in different lineages simplifiable into a Boolean-logic function (A and B and C), which is implementable in a single cell. Using this minimization scheme, we minimized the required number of strains from six to two.

We created an algorithm in Python to automate this simplification. We generated all Boolean functions and converted each truth table into a lineage tree. For the implementation of history-dependent programs, we tested if any Boolean functions can be extracted from this program. If the use of Boolean devices leads to an implementation with an equal or reduced number of strains, the design is saved. We then obtained as output a list of designs based on Boolean and/or history-dependent devices implementing the input program with the minimal number of strains possible. We applied this brute-force method to all 3-input/1-output programs, totaling 65,536 programs. This strategy allows for a reduction in the number of strains for the implementation of 20% of these programs. It does not significantly reduce the median number of strains required for the implementation of history-dependent programs, as shown in Figure 3.5B. However, 48% of the 3-input/1-output programs are decomposable using Boolean programs while minimizing the number of strains (Table 3.1, Figure 3.6).

As we only automated the design of single-output Boolean functions, Boolean logic devices can only be used to implement 1-output sub-programs, which reduce their use for the implementation of multi-output history-dependent programs. Contrarily, multi-output Boolean functions could be implemented with history-dependent devices.

Our Python algorithm is also applicable to 4- and 5-input history-dependent programs. Ac-

# strains	Without minimization	With minimization	Difference of strains with and without minimization for each program
0			53080
1	79	99	10779
2	1122	1714	1050
3	7202	9922	461
4	20196	23334	157
5	25272	23298	8
6	11664	7168	**31898 useful to use Boolean logic devices**

Table 3.1: Number of programs requiring a specific number of strains for implementation with and without minimization using Boolean devices.

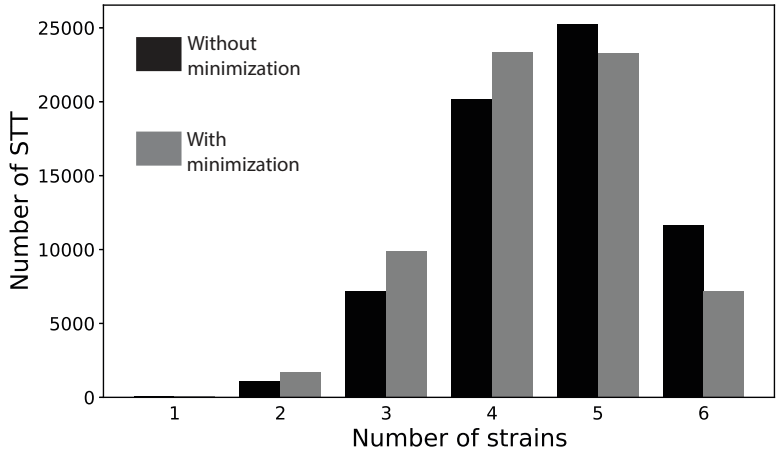


Figure 3.6: **Distribution of the number of strains required for the implementation of all history-dependent programs.** Y-axis represents the number of programs requiring a specific number of strains for implementation (x-axis), for all 3-input 1-output history-dependent programs. Black bars correspond to the data without simplification with Boolean logic devices and the grey bars with simplification. Data were obtained using a Python algorithm which generated the designs with the various strategies for all programs.

cording to the large number of 4- and 5-input history-dependent programs, we did not generate all programs to quantify the minimization capacity of this strategy to 4 and 5 inputs. As using brute-force strategy the computation time would increase exponentially.

The implementation of history-dependent programs could also be minimized by decomposition of programs into sub-programs integrating fewer inputs (Figure 3.7). A systematic minimization algorithm exists for Boolean functions, but there is not currently one for history-dependent programs. Minimization would be possible using brute-force strategy like for the decomposition of programs in Boolean and history-dependent programs. However, this strategy would not be possible for an increasing number of inputs. A systematic method would have to be defined for the minimization of history-dependent programs.

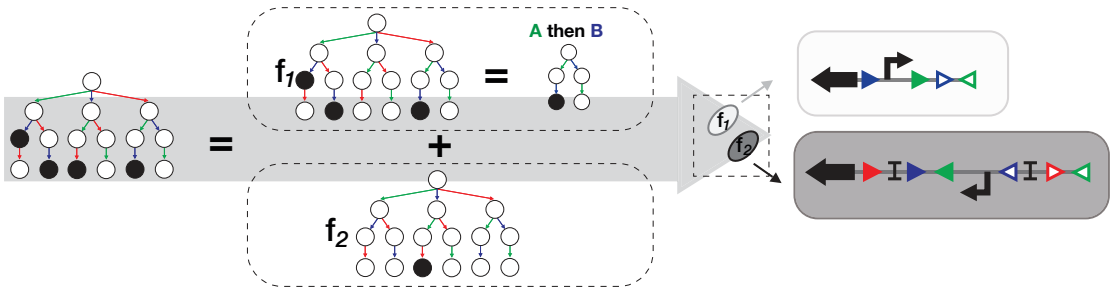


Figure 3.7: **Minimization of history-dependent programs by decomposition into programs of a reduced number of inputs.** Example of a 3-input program (initially decomposed in three lineages) which can be minimized in programs with a reduced number of inputs. The program can be decomposed in two sub-programs. The first program corresponds to a lineage tree with three ON states in different lineages (normally implemented in three different strains). This program is simplifiable in a 2-input lineage (A then B) implementable in a single strain. Using this simplification, the complete program initially implemented in three strains is implemented in two using a 2-input and 3-input history-dependent logic device.

3.3 Implementation of history-dependent gene-expression programs in multicellular consortia

As a proof of concept of our history-dependent logic design, we implemented 2- and 3-input gene-expression programs in the gram-negative model bacteria *Escherichia coli*.

3.3.1 OSiRIS: Optimization by SynthesIs of Recombination Intermediate States

Using irreversible recombinase switches, the different states of our history-dependent system are encoded in DNA. To simplify the optimization of the 2- and 3-input history-dependent scaffolds (Figure 3.2A and 3.3A), we synthesised and characterized the different recombination intermediate states. We then optimized the history-dependent devices independently to the sequential integrase switches. We called this optimization workflow OSiRIS for Optimization by Synthesis of Intermediate Recombination States.

In the OSiRIS workflow, we first designed the history-dependent device with expression of a different fluorescent gene in each input state (Figure3.8A). Then, the DNA sequences of the different recombination intermediate states were generated and synthesized. Each construct was characterized by quantification of the fluorescence intensity in the different channels and was compared to the expected phenotype. If the phenotype did not match, the multi-output scaffold was redesigned and a new OSiRIS cycle was performed. This approach permits to accelerate the optimization of history-dependent devices.

3.3.1.1 Validation of the 2-input scaffold design via OSiRIS

We first applied the OSiRIS workflow to a 2-input history-dependent lineage. For 2-input history-dependent programs, four different recombination states correspond to the five input states: the original state (in absence of input) and three intermediate states (Figure3.8B). Two different input states (input B, input B then input A) result in the same DNA intermediate state.

For the design of the 2-input multi-output scaffold, we implemented the A then B lineage while associating Bxb1 integrase to the input A and Tp901 integrase to the input B (Figure 3.9A). We selected the fluorescent proteins sfGFP, mKate2, and BFP, as their excitation and emission spectrums do not overlap. We used P6 as the promoter and B0034 as the ribosome binding site. To insulate the translation from the genetic context, we placed a ribozyme in 5' end of each output gene, catalyzing the cleavage of the mRNA at this position [Lou 2012]. We used different ribozymes for each output gene (RiboJ, BydvJ, and AraJ) to avoid multiple repetitions of sequences in the construct. Finally, we added 40 bp spacers designed for Gibson assembly: sp0, sp4, sp5, and spN.

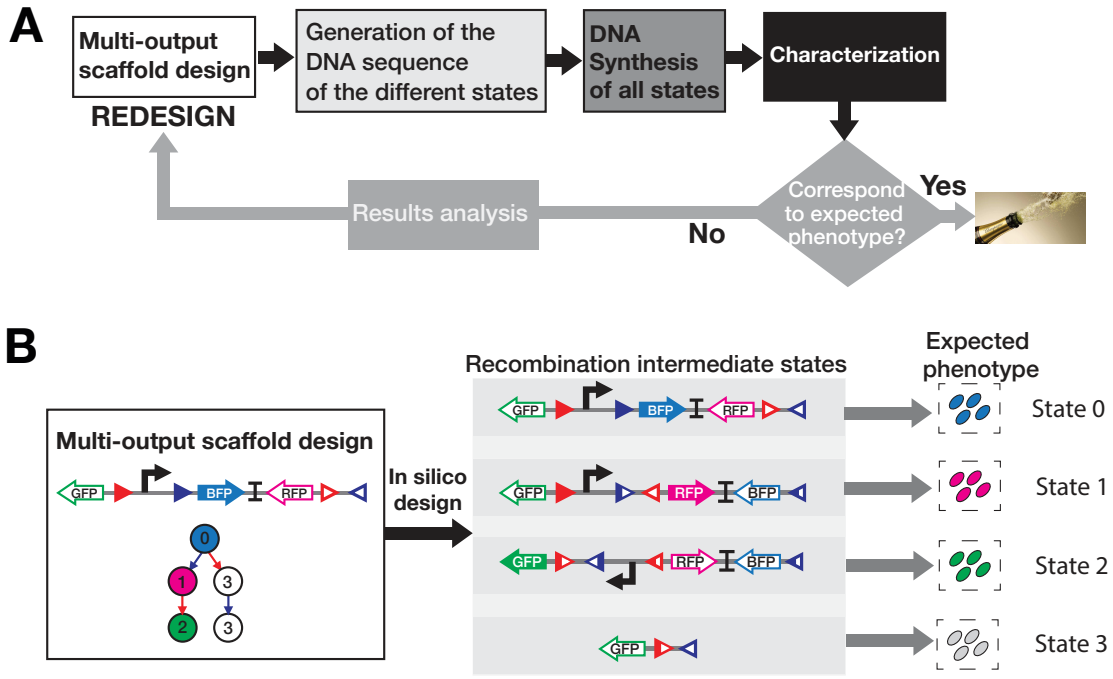


Figure 3.8: **Optimization by SynthesIs of Intermediate Recombination States.** (A) OSiRIS workflow. A history-dependent scaffold corresponding to a specific lineage and with expression of a different gene in each input state is designed. The DNA sequences of the different input states corresponding to the intermediate recombination states are generated. The initial and the intermediate sequences are synthesized and characterized, and the phenotypes are compared to the expected phenotypes. If they match, a celebration is performed; otherwise, the results are precisely analyzed to identify the origin of the bug, the multi-output scaffold is redesigned, and a new OSiRIS cycle is performed. (B) 2-input OSiRIS workflow. For 2-input, a scaffold with consecutive expression of BFP, RFP, and GFP in the three lineages strain is designed. From this design, the 3 intermediate recombination states are generated and the expected phenotype for each of the four states is predicted.

Based on this designed 3-output/2-input history-dependent device, we generated, synthesized and constructed the mother construct and the three intermediate recombination states. We characterized the four constructs by flow cytometry and microscopy (Figure 3.9B). For each DNA state, we obtained the expected phenotype: BFP in state 0, RFP in state 1, GFP in state 2, and no expression in state 3. No further optimization was required for this 2-input history-dependent scaffold.

3.3.1.2 Optimization of the 3-input scaffold via OSiRIS

Based on the previously characterized 2-input scaffold, we designed a 3-input scaffold by adding a third integrase, Int5, and a new output gene, LacZ alpha (Figure 3.10A). Therefore, the 3-input scaffold implements the lineage: input C with Int5 then input A with Bxb1 then input B with Tp901. The use of Int5 as the first input permits us to keep the same backbone as that for the 2-input scaffold. Following the OSiRIS workflow, we generated, synthesized, and characterized

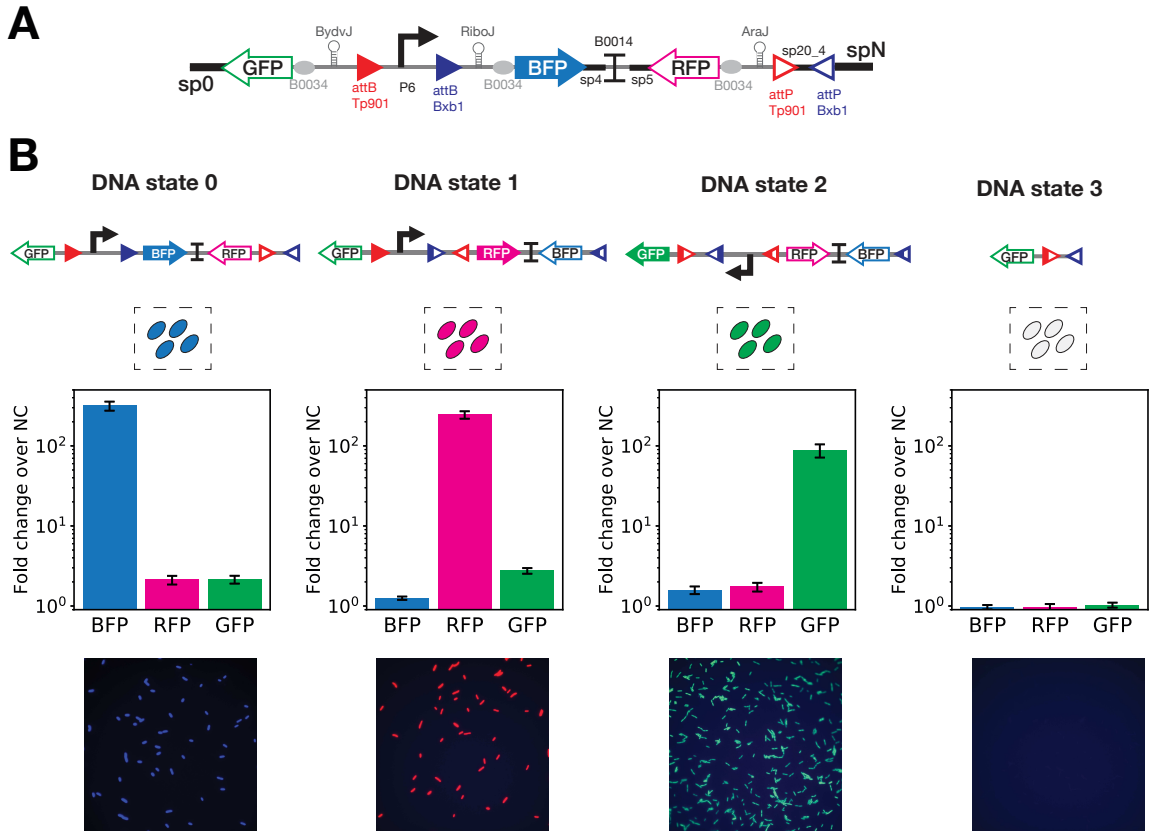


Figure 3.9: **Design and characterization of 2-input OSiRIS.** (A) Detailed design of the 2-input scaffold for the lineage A then B with Bxb1 for input A and Tp901 for input B. As output genes, we used BFP, RFP, and GFP. In the 5'UTR of each gene we placed a ribozyme and the RBS B0034 to isolate translation from genetic context. We used P6 as the promoter and B0014 as a bidirectional terminator between BFP and RFP coding sequences. We added 40 bp spacers for Gibson assembly (sp0, sp4, sp5, and spN) and a 20 bp spacer between to juxtaposed integrase sites (sp20_4). (B) Characterization of the 2-input OSiRIS by flow cytometry and microscopy. We characterized each initial and intermediate recombination state via flow cytometry by measurement of GFP, RFP, and BFP fluorescence intensity. The bar graph corresponds to the fold change over the negative control (strain without fluorescent protein) for each channel from two experiments with three replicates per experiments (detailed in Materials and methods). The error bars correspond to the standard deviation between the fold change obtained in the two separated experiments. The microscopy images correspond to merged images of the GFP, RFP, and BFP channels.

the 3-input scaffold and the five recombination intermediate states. Unfortunately, we did not obtain the expected phenotypes (Figure 3.10B). For the DNA state 0, 1, and 4, GFP was expressed at about 25 times above the negative control when no GFP expression was expected. Moreover, for the DNA state 3, RFP was expressed at eight times above the negative control, while no RFP expression was expected. Otherwise, BFP, RFP, and GFP fluorescent proteins were expressed at the expected DNA states. By comparison with DNA states from the 2-input scaffold, we supposed that the unexpected expression of GFP or RFP was due to the gene expression cassette of LacZ alpha. One possibility is that the terminator L3S2P21 is not strong

enough to stop transcription or that the spacer 7 is promoting transcription as even without a promoter the GFP gene was still expressed (DNA state 4).

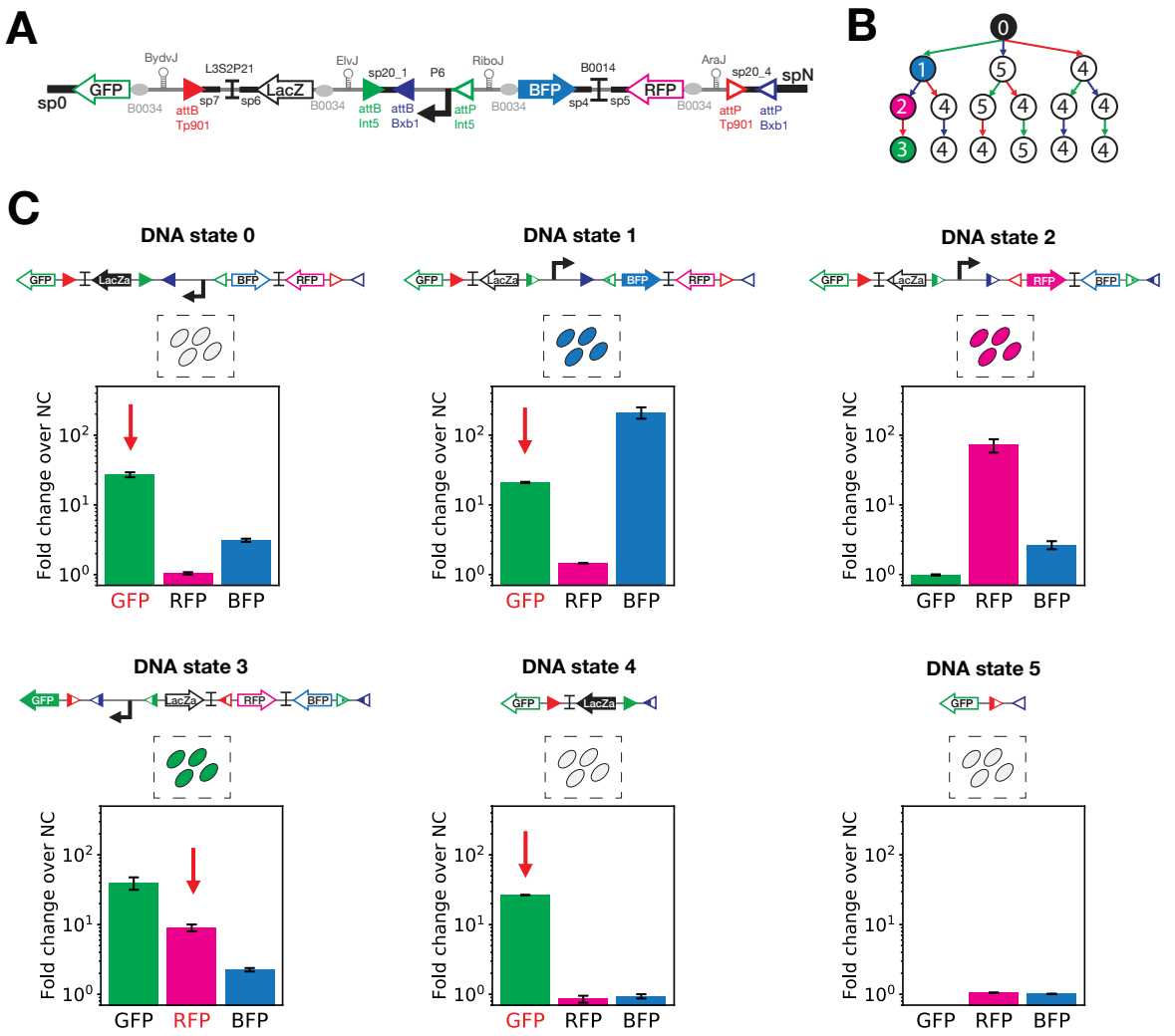


Figure 3.10: **Design and characterization of the first version of 3-input OSiRIS.** (A) Detailed design for the first version of the 3-input scaffold for the lineage C then A then B with Bxb1 for input A, Tp901 for input B, and Int5 for input C. As output genes, we used LacZ alpha, BFP, RFP, and GFP and in the 5'UTR of each gene we placed a ribozyme and the RBS B0034 to insluate translation from genetic context. We used P6 as a promoter and L3S2P21 and B0014 as bidirectional terminators. We added 40 bp spacers for Gibson assembly (sp0, sp4, sp5, sp6, sp7, and spN) and a 20 bp spacer between juxtaposed integrase sites (sp20_1, sp20_4). (B) 3-input lineage tree corresponding to the designed 3-input scaffold. The color of each node corresponds to the expected phenotype and the number in the node to the corresponding DNA state of each input state. (C) Characterization of the 3-input OSiRIS by flow cytometry. We characterized each initial and intermediate recombination states via flow cytometry by measuring GFP, RFP, and BFP fluorescence intensity. The bar graph corresponds to the fold change over the negative control (strain without fluorescent protein) for each channel from two experiments with three replicates per experiments (detailed in Materials and Methods). The error bars correspond to the standard deviation between the fold change obtained in the two separated experiments.

Taking advantage of our OSiRIS workflow, we designed two new versions of the 3-input scaffold (Figure 3.11A). For both versions, we replaced the L3S2P21 terminators by L3S3P21 and J61048 and removed the LacZ alpha operating unit for simplification purpose. We designed one version with sp7 and sp6 spacers and one without a spacer. We then characterized two representative states for each version: (1) the initial state and (2) the DNA state 4, as we previously observed GFP expression while not having a promoter. The version 1 corresponds to the version previously characterized. For version 2 (with sp7 and sp6 spacers), we obtained, as with the original version, GFP expression in the two characterized states (Figure 3.11B). For version 3, we obtained the expected phenotype, no significative expression of GFP, RFP, or BFP fluorescent proteins in the initial state or in DNA state 4 (Figure 3.11B). Consequently, we supposed that the GFP expression in versions 1 and 2 were due to promoter activity of spacer 7.

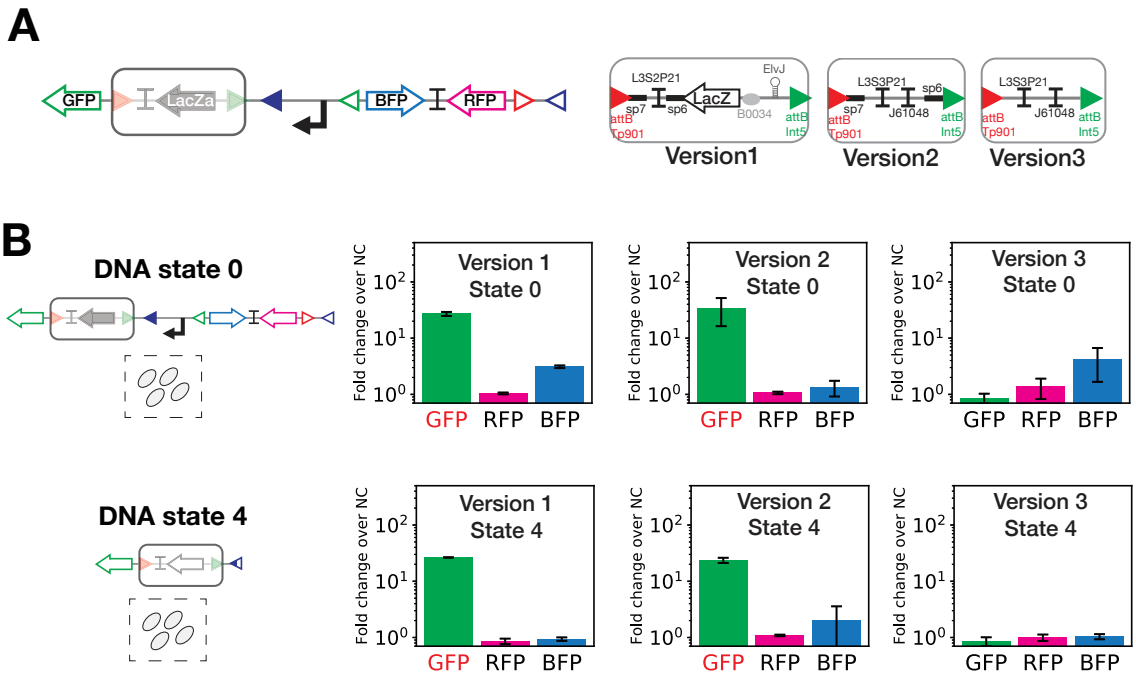


Figure 3.11: **Optimization of 3-input OSiRIS.** (A) Generation of two new 3-input OSiRIS design by modification of the cassette between the attB Tp901 integrase site and the attB int5 site. Version 1 corresponds to the previously described design. For the second and third version, the LacZ gene and 5'UTR were removed and the L3S2P21 terminator was replaced by the two terminators L3S2P21 an J61048. For version 3, we also removed the two spacers sp7 and sp6. (B) Characterization of the initial state and DNA state 4 of the three versions of the 3-input OSiRIS by flow cytometry. We characterized the two selected states via flow cytometry by measuring GFP, RFP, and BFP fluorescence intensity. The bar graph corresponds to the fold change over the negative control (strain without fluorescent protein) for each channel from two experiments with three replicates per experiment (detailed in Materials and Methods). The error bars correspond to the standard deviation between the fold change obtained in the two separate experiments. The red labeling corresponds to constructs which did not behave as predicted.

3.3.1.3 Validation of the optimized 3-input scaffold via OSiRIS

We then selected as a final 3-input scaffold version 3 of the design. We constructed and characterized each remaining recombination intermediate state, which expressed the expected phenotype for each DNA state (Figure 3.12).

Version 3 of the design did not express any output gene in the initial input state. Therefore, we additionally designed a device with expression of LacZ alpha in the first input state to test the feasibility of expressing an output gene in each lineage state. By adding X-gal to the media, we obtained a clear blue coloration of bacteria with this construct and no coloration for bacteria in intermediate states.

To summarize, we optimized the design of a 4-output/3-input history-dependent scaffold. The OSiRIS workflow allowed us to efficiently optimize this large logic device.

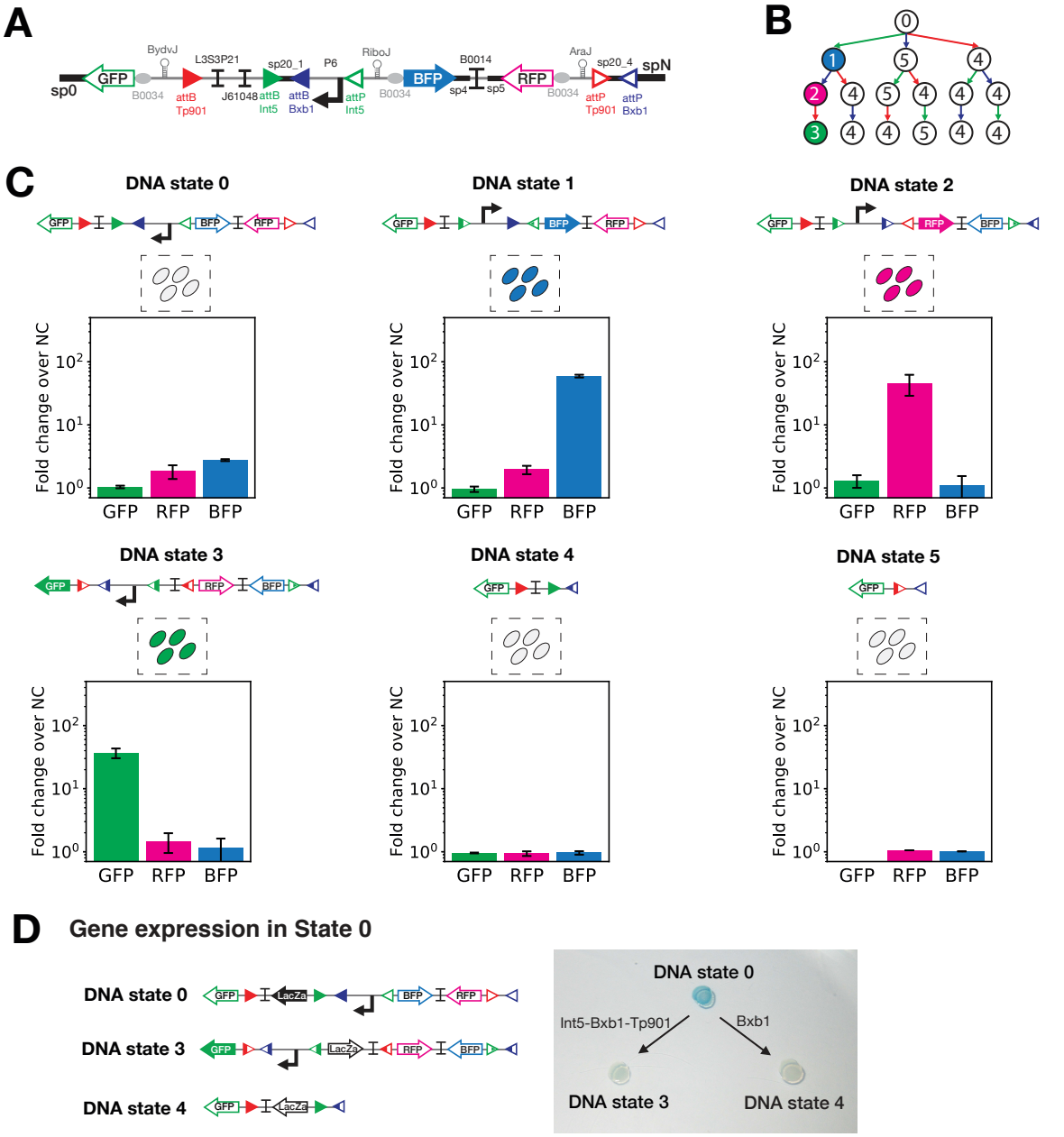


Figure 3.12: **Final design and characterization of the optimized 3-input OSiRIS.** (A) Detailed design for the final 3-input scaffold. In this design, no gene is expressed in the initial input state. Otherwise the design corresponds to the previously described design. (B) 3-input lineage tree corresponding to the final 3-input scaffold. The color of each node corresponds to the expected phenotype and the number in the node to the corresponding DNA state in each input state. (C) Characterization of the final 3-input OSiRIS by flow cytometry. We characterized each initial and intermediate recombination states via flow cytometry by measurement of GFP, RFP, and BFP fluorescence intensity. The bar graph corresponds to the fold change over the negative control (strain without fluorescent protein) for each channel from three experiments with three replicates per experiment (detailed in Materials and Methods). The error bars correspond to the standard deviation between the fold change obtained in the three separate experiments.

3.3.2 Characterization of a history-dependent program by sequential induction

3.3.2.1 Characterization of a single-cell 2-input program

For characterization of a full 2-input history-dependent system with inducible integrases, we used the dual-controller plasmid from Bonnet et al. [Bonnet 2013] as a sensing device. The dual controller permits induction of Bxb1 integrase by aTc (Anhydrotetracycline) and Tp901 by arabinose. Therefore, aTc corresponds to the input A and arabinose to the input B. We co-transformed the dual controller plasmid with our 2-input scaffold. We worked in "fundamental mode", considering that inputs do not occur simultaneously but sequentially. We then performed sequential inductions: a first overnight induction for the first input followed by second overnight induction for the second input. We characterized the phenotype in each induction condition after three days by flow cytometry. In each input state, we obtained the expected phenotype: expression of BFP in absence of induction, RFP with aTc only, GFP with aTc on the first day and Arabinose on the second day, and no expression otherwise (Figure 3.13).

In conclusion, we were able to implement a 2-input 3-output history-dependent program in *Escherichia coli*.

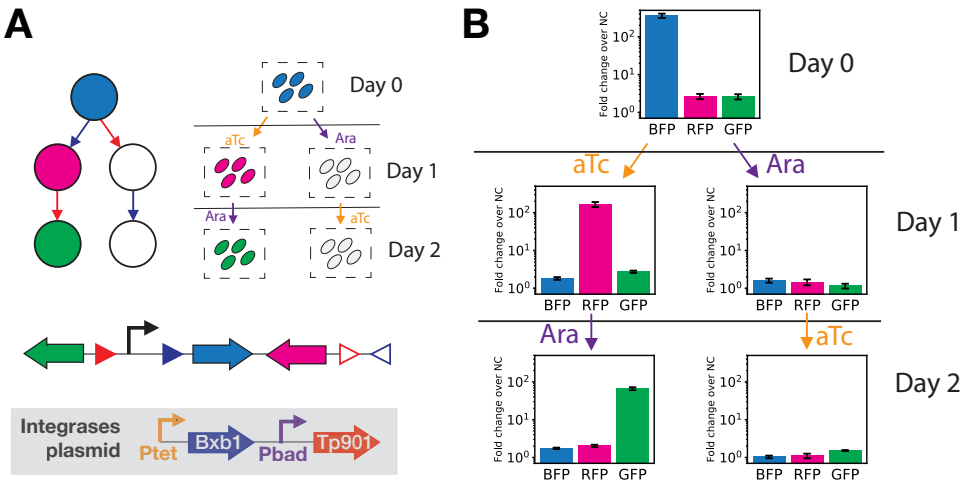


Figure 3.13: **Characterization of a 2-input/3-output history-dependent program by sequential induction.** (A) Experimental setup of a 2-input history-dependent program. We co-transformed the previously characterized 2-input scaffold with a plasmid for inducible expression of Bxb1 and Tp901 integrase. Bxb1 expression is induced by aTc (input A) and Tp901 by arabinose (input B). The lineage tree of this program and corresponding expected cell behaviors are represented. (B) For characterizating the system, the co-transformed bacteria are induced twice for 16 hours each. Each graph corresponds to a different induction condition corresponding to an input state of the lineage tree. The bar graph corresponds to the fold change over the negative control (strain without fluorescent protein) for each channel (GFP, RFP, BFP) from three experiments with three replicates per experiment. The error bars correspond to the standard deviation between the fold change obtained in the three separate experiments. (More details in Material and Methods.)

3.3.2.2 Characterization of a multi-cell 2-input program

The implementation of 2-input history-dependent programs can require the combination of up to two strains corresponding to the two lineage subprograms. To prove the feasibility of a multicellular system, we designed a 2-strain 2-input program (Figure 3.14). We designed and synthesized the two devices required for the implementation of the two sub-programs (Figure 3.14A). We based this design on the previously characterized 2-input scaffold. Because each device implements a different lineage, the position of integrase sites are inverted.

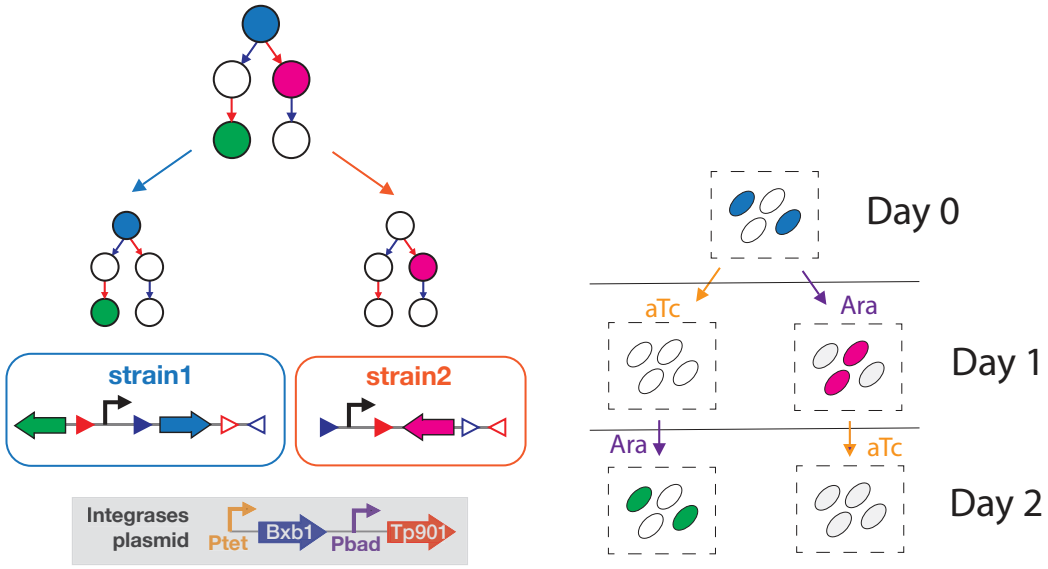


Figure 3.14: **Design and expected phenotype of a 2-input history-dependent program requiring composition of 2 strains.** The program corresponds to the expression of BFP in absence of input, expression of RFP in presence of input B only, and GFP in presence of input A then input B. The program is decomposed in two sub-programs, each implemented with one logic device in a separated cell. Strain 1 implements the lineage A then B with expression of BFP in absence of input and expression of GFP in presence of A then B. Strain 2 expresses the lineage B then A with expression of RFP in presence of B only. Between the logic devices, the position of integrase sites are switched to implement the two different lineages. Each logic device is co-transformed with the dual-controller plasmid and strains are grown together. We expected to have half of the bacterial population express any fluorescent output (expected cell behavior in the right panel).

We then characterized this program in *Escherichia coli*. We characterized the two different strains separately and as a multicellular system (Figure 3.15). For all experiments, we co-transformed the two history-dependent devices with the dual controller. For the individual characterization, the sequential induction was performed as previously. For characterizing the multicellular system, the two strains were mixed in a co-culture after overnight growth in stationary phase. We characterized the fluorescent profile of the individual strains and the multicellular culture in bulk using a plate reader. For all characterization, we obtained the

expected fluorescent profile. For strain 1, we obtained the highest background expression of GFP in the input states with aTc only and Arabinose only when we did not expect expression. As these data correspond to a single experiment, we cannot conclude anything for now. This experiment is being replicated.

In the characterization of the multicellular system, we expected to obtain a 2 fold decrease of the output fluorescent expression as half of the population is not expressing the output fluorescent protein. For BFP expression we obtained a 7-fold decrease, for RFP expression a 1.6-fold decrease and for GFP expression a 1.5-fold decrease. The results for BFP expression are surprising; consequently, the analysis of the percentage of each population will be performed via flow-cytometry. These results are encouraging and need to be confirmed by future experimental replicates.

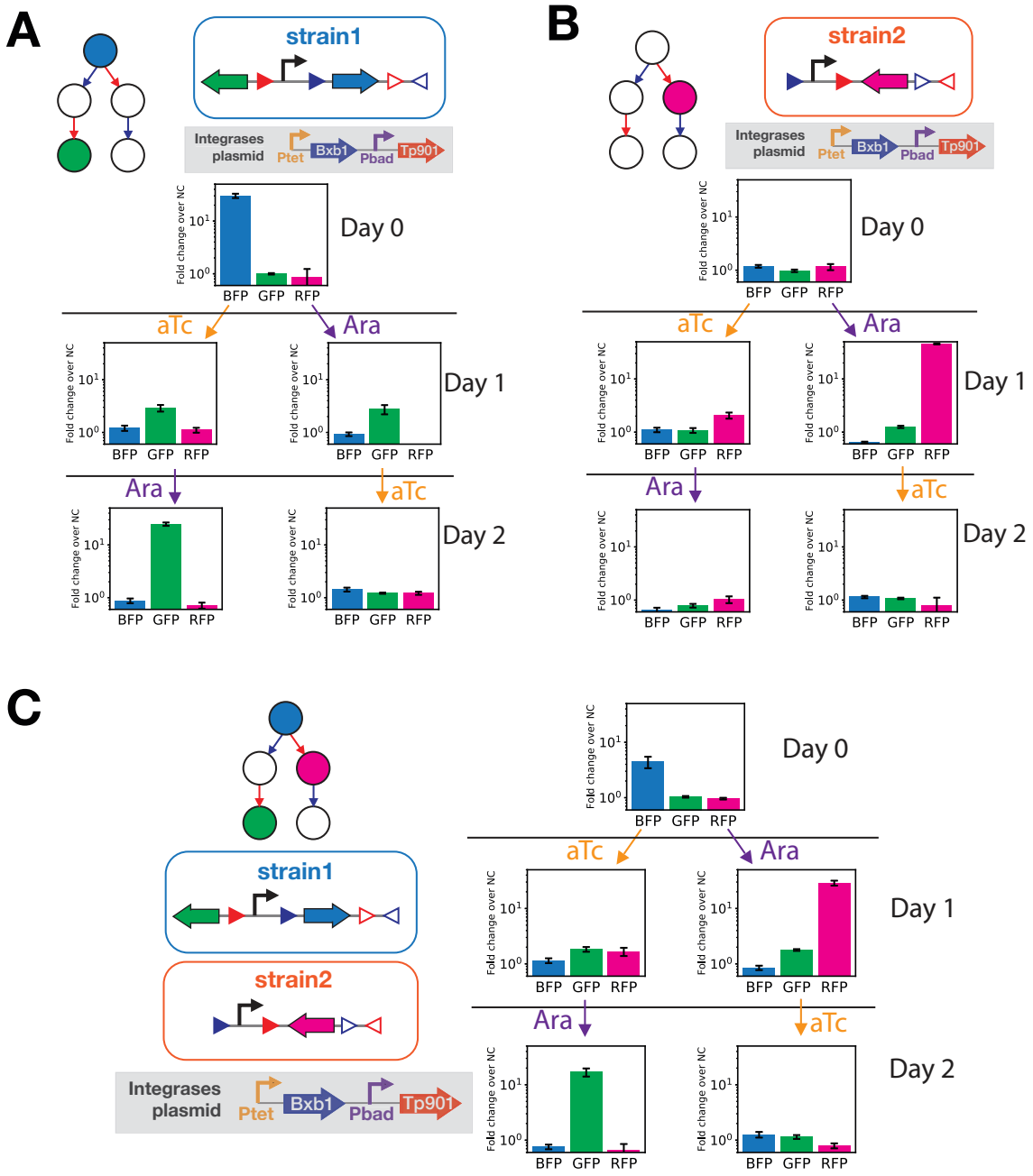


Figure 3.15: **Characterization of a multicellular 2-input history-dependent circuit.** (A) (B) For each strain, the lineage tree implemented by the strain is represented next to the design of the corresponding device. We separately co-transformed the two history-dependent devices with the dual-controller plasmid for inducible expression of Bxb1 and Tp901 integrase. The strains are then induced twice 16 hours each (more details on the Materials and Methods). Each graph corresponds to a different induction condition corresponding to an input state of the lineage tree. The bar graph corresponds to the fold change over the negative control (strain without fluorescent protein) for each channel (GFP, RFP, BFP) from one experiments with three replicates per experiment (detailed in Materials and methods). The error bars correspond to the standard deviation between the fold change of the three replicates.

3.3.2.3 Design for the characterization of a single-cell 3-input program

For the characterization of the 3-input/4-output history-dependent programs, we are currently lacking an inducible integrase. Indeed, integrases have to be precisely tuned to avoid expression leakage and unexpected switches. For this purpose, we are developing a method to automate the connection between inducible promoters and integrases. We are currently connecting the PyeaR promoter (responding to nitric oxide) with Int5 integrase. This additional inducible integrase will permit us to characterize our optimized 3-input/4-output history-dependent device via sequential induction over four consecutive days (Figure 3.16).

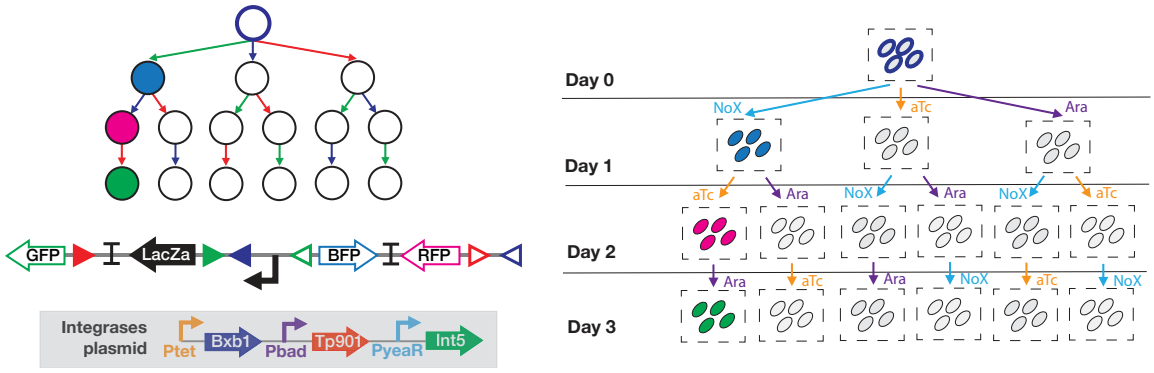


Figure 3.16: **Characterization workflow of the 3-input/4-output history-dependent program.**

3.4 Discussion

Coupling distributed multicellular computation (DMC) with the memory of integrase switches, we implemented history-dependent programs in living organisms.

We developed an automated framework for designing of up to 5-input and N-output history-dependent gene-expression programs. In comparison to previous integrase-based history-dependent designs, we used simple logic devices based on a single integrase and single pair of sites per input. Nevertheless, DMC allows the design of complex circuits from simple, elementary building blocks. The design of complex circuits is accessible using our framework. Consequently, we provided a web-interface for the design of all 5-input history-dependent gene-expression programs, which improves significantly upon previous systems that used brute-force computation methods for the design of 3-input history-dependent programs.

The main limitation of our design is the high number of strains required for the implementation of some history-dependent programs. Increasing the number of strains in the system will lead to difficulty in detection of the output expression from one strain and can lead to growth competition between strains. One solution to reduce the competition between strains would be to encapsulate strains in alginate beads. Moreover, as the output state is encoded within DNA, high-throughput DNA-sequencing methods could be used to read the output of the system. With this method, cells in small proportion would still be detectable.

To reduce the number of required strains, we proposed several simplification strategies. First, we developed a brute-force minimization algorithm for combining history-dependent and Boolean logic devices. Using brute-force computation method, this minimization strategy is applicable for up to four inputs, the computation time is too lengthy for five inputs. Nevertheless, it allows for the reduction of the number of required strains by 20% for 3-input programs. Additionally, a more important minimization would be accessible by developing an algorithm for systematic minimization and decomposition of history-dependent programs similar to the one for Boolean function. This would permit to decompose programs in sub-programs with a reduced number of inputs.

In addition to this automated design workflow, we implemented history-dependent gene-expression programs in *Escherichia coli*. The implementation is based on a scaffold corresponding to the implementation of one lineage. All programs corresponding to one lineage are accessible based on this scaffold by placing an output gene at the GOI position(s) with an ON input state(s). All programs are implementable based on this scaffold design. As the state of the system is encoded in DNA, we developed the OSiRIS workflow for scaffold optimization by synthesis of each intermediate recombination states. The OSiRIS workflow permits a characterization independent of the input and integrase switches. It allowed us to identify the cause of gene-expression leakage for the first version of the 3-input scaffold and to efficiently optimize this design. Without using OSiRIS, it would have been difficult to distinguish the cause of

this expression between a possible leakage in the integrase expression, an incomplete integrase switch, or a problem with gene expression in the device. Using the OSiRIS workflow, we will construct, characterize, and optimize a 4-input scaffold. As five genes are needed for the 4-input scaffold, we will design the scaffold by alternating expression of the three fluorescent protein RFP, GFP, and BFP (e.g. RFP-GFP-BFP-RFP-GFP). The use of a different fluorescent protein for each output state is not possible as there are not currently five fluorescent proteins with compatible excitation and emission spectrums.

Based on the optimized 2-input scaffold, we characterized a full history-dependent program with the dual-controller plasmid for integrase inductions. The system worked as predicted with complete switches in each state and clear output fluorescent intensity in each ON state. We are currently characterizing a multicellular 2-input program following the same design strategy. Moreover, we are working in collaboration with Pascal Hersen and Zacchary Ben Meriem (Laboratoire Matiere et Systemes Complexes, Paris Diderot) to obtain video of time-lapse induction of our 2-input/1-strain history-dependent system. To do so, we used a mother machine microfluidic system. The first results are promising and we are now optimizing the imaging condition. Based on these preliminary results, three hours of induction appear to be sufficient to have a complete switch. Further characterization of the system should be perform to determined the minimal time required between two inputs to avoid a mixed output population.

For the full 3-input system implementation, we are now engineering a third integrase switch, PyeaR with Int5 integrase. With this third switch, we will complete the implementation of the 3-input history-dependent program (Figure 3.16). Another switch responding to benzoic acid is also promising.

Moreover, as detailed previously, minimized systems are accessible by a combination of history-dependent and Boolean logic devices. As we previously characterized all 4-input logic devices, we will implement history-dependent programs by combining existing 3-input Boolean and history-dependent devices. This combination highlights the interest of distributing multicellular computation as each strain can be designed using a completely different strategy.

3.5 Materials and Methods

3.5.1 Equations for the determination of number of functions/strains/devices for history-dependent logic

History-dependent programs are represented as a lineage tree. Each node of this tree corresponds to a specific state of the system in response to a different scenario: when no input occurred, when one input occurred, and when multiple inputs occurred in a particular sequence. For an N-input program, the number of states is equal to (eq.1).

$$Number\ _{states} = \sum_{k=0}^{N} \frac{N!}{k!} (eq.1)$$

Then, for N-input/1-output history-dependent logic programs, the number of possible programs is equal to 2 to the power of the number of states (eq.2), as all states can have either a ON or OFF output. Similarly for N-input/M-output history-dependent logic programs, 2 to the power of the number of states multiplied by M programs exist (eq.3).

$$Number\ _{1-output\ programs} = 2^{Number_{states}} = 2^{\sum_{k=0}^{N} \frac{N!}{k!}} (eq.2)$$

$$Number\ _{M-outputs\ programs} = 2^{M.Number_{states}} = 2^{M \sum_{k=0}^{N} \frac{N!}{k!}} (eq.3)$$

The maximum number of strains needed to implement an N-input/M-output history-dependent gene-expression program is equal to N factorial, which corresponds to the number of possible lineages in an N-input lineage tree.

3.5.2 Automated generation of genetic designs to execute multicellular Boolean logic and history-dependent gene expression programs

We encoded an algorithm capable of creating up to 5-input history-dependent program designs using Python (Figure 3.4A). The algorithm takes as input a lineage tree (equivalent to a sequential truth table). The output corresponds to the biological implementation, such as a graphical representation of the genetic circuit and its associated DNA sequences for each strain.

The lineage tree is decomposed into sub-trees consisting of a single lineage containing one or multiple ON states. This decomposition is done by iteratively subtracting the lineages containing ON states. To obtain the lowest number of sub-programs, we prioritize among the lineages with ON states the ones for which the highest number of inputs occurred (from the right to the left of the lineage tree). After decomposition, for each selected lineage, two pieces of information are extracted. First, based on which states are ON, we directly design the

corresponding scaffold by specifically inserting genes at the adequate GOI positions. Second, the order-of-occurrence of inputs corresponding to the lineage is used to identify which sensor modules are needed among the different connection possibilities between control signals and integrases. Then, by combining the design of the different lineages, we obtain the global design for biological implementation of the desired history-dependent gene-expression program.

To simplify the construction process of logic circuits, DNA sequence of computation devices is generated by our Python code. In CALIN, sequences are adapted for *E.coli*. But sequence generation can be adapted to other organisms (databases are available for *B. subtilis* and *saccharomyces cerevisiae*) or customly designed using the source Python code available on Github.

As these methods are straightforward, they support the generation, in a reduced time, of biological designs performing complex programs in response to a large number of inputs.

3.5.2.1 Construction and characterization of 2-input and 3-input OSiRIS

For *E. coli* strains, media, and molecular biology procedures please refer to the Materials and Methods of Chapter 2.

2- and 3-input OSiRIS constructions

As the constructs are large, most of them were divided in multiple DNA fragments and ordered via Twist. Multiple fragment Gibson assemblies were then performed. As vectors, either pSB4K5 with sp0 and spN or previously cloned constructs were used. For each construct, we used different DNA fragment(s) and PCR amplified vectors; all the information is listed in the following table.

Description	Name	Vector (P1-P2)	DNA fragments
2-input scaffold	F0	pSB4K5 (P71-P72)	Gb52-Gb53-Gb54
2-input state 1 OSiRIS	A2	pSB4K5 (P71-P72)	Gb52-Gb73-Gb74
2-input state 2 OSiRIS	A3	pSB4K5 (P71-P72)	Gb75-Gb76-Gb74
2-input state 3 OSiRIS	A4	pSB4K5 (P71-P72)	Gb77
3-input scaffold v1	A5	F0 (P71-P870)	Gb78-Gb79-Gb80
3-input state 1 OSiRIS v1	A6	F0 (P71-P870)	Gb78-Gb81-Gb82
3-input state 2 OSiRIS v1	A7	pSB4K5 (P71-P72)	Gb78-Gb81-Gb73
3-input state 3 OSiRIS v1	A8	pSB4K5 (P71-P72)	Gb75-Gb81-Gb84-Gb83
3-input state 4 OSiRIS v1	A9	pSB4K5 (P71-P72)	Gb78-Gb85
3-input state 5 OSiRIS v1	A10	pSB4K5 (P71-P72)	Gb77
3-input scaffold v2	B7	A5 (P1366-P1319)	Gb104
3-input state 4 OSiRIS v2	B8	A9 (P1366-P72)	Gb105
3-input scaffold v3	B5	A5 (P1323-P1319)	Gb102
3-input state 4 OSiRIS v3	B6	A9 (P1323-P72)	Gb103
3-input state 1 OSiRIS v3	B21	A6 (P1323-P1318)	Gb110
3-input state 2 OSiRIS V3	B22	A8 (P1322-P1829)	Gb111
3-input state 3 OSiRIS v3	B23	A8 (P1323-P1829)	Gb112
3-input LacZ scaffold v3	B24	A5 (P1323-P1832)	Gb113
3-input LacZ state 4 OSiRIS v3	B28	A9 (P1323-P1832)	Gb113

Table 3.2: Cloning information of OSiRIS constructs.

OSiRIS characterization

For the characterization of the OSiRIS constructs, the same protocol as for part characterization in Chapter 2 was used, with the only difference being the medium. For all history-dependent device characterization, Hi-Def Azure medium (purchased from Technova) supplemented with 0.4% of glycerol was used.

For these characterizations, except for Figure 3.12, three fluorescent channels were analyzed via flow cytometry. GFP fluorescence intensity was measured by excitation with a 488 nm laser and a 510/10 nm filter (BL1). RFP excitation was performed by a 561 nm laser and filter 615/25 nm (YL2). BFP excitation was performed by a 405 nm laser and filter 440/50 nm (VL1). As detailed in Chapter 2, the data was analysed using Flow-Jo.

For the characterization of the third version of 3-input OSiRIS, the measurement was performed via plate reader using a BioTeck Cytation 3. Cultures were diluted four times in PBS and measured with the following parameters (GFP: excitation 485 nm, emission 528 nm, gain 80, BFP: excitation 402 nm, emission 457 nm, gain 70, RFP: excitation 555 nm, emission 584 nm, gain 100, absorbance: 600 nm). For each sample, GFP, BFP, and RFP fluorescence intensity normalized to absorbance at 600nm were calculated and the mean value was calculated between the three replicates. The fold change over the negative control was determined from this mean value over that of the negative control. The mean fold change was represented in the figure corresponding to the mean of the fold change of the three experiments. The error bars correspond to the standard deviation between the three experiments.

3.5.2.2 Construction and characterization of 2-input history-dependent programs

Construction for multi-cell 2-input history-dependent programs

Two additional history-dependent devices were constructed for the multi-cell characterization. Following the amplified vectors and DNA fragments were used to perform the Gibson assembly of these two constructs.

Description	Name	Vector (P1-P2)	DNA fragments
2-input history-dependent device BFP-0-GFP	B1	F0 (P38-P72)	Gb98
2-input history-dependent device 0-RFP-0	B2	pSB4K5 (P71-P72)	Gb99

Table 3.3: Cloning information of OSiRIS constructs.

2-input history-dependent program single-cell characterization

For the 2-input history-dependent program characterization in single-cell, the history-dependent devices (such as F0, B1, and B2) were co-transformed with the dual controller in J64100 [Bonnet 2013]. For transformation and further culture, Hi-Def media were supplemented

with 12.5 μg/mL of kanamycin and 25 μg/mL of carbenicillin.

As a control, the OSiRIS constructs (F0, A2, A3) and a strain without fluorescent protein were used. For the first day of characterization, 96 deep-well plates filled with 500 μL per well of Hi-Def Azure supplemented with glycerol were inoculated with three clones per co-transformation and three clones per control. Plates were grown for 16 hours at 37°C. Cultures were diluted 40 times on Focusing Fluid and directly measured on a flow cytometer according to previously described methods (as for OSiRIS characterization). For the first induction, each culture from the co-transformation was diluted 1, 000 times in fresh HI-Def Azure-Glycerol, with no inducer, 1% of arabinose, or 200 ng/mL of aTc. Plates were grown for 16 hours at 37°C. Cultures were diluted 40 times with Focusing Fluid and directly measured on a flow cytometer. For the second induction, each culture from the previous induction was diluted 1, 000 times in fresh HI-Def Azure-Glycerol, with no inducer, 1% of Arabinose, or 200 ng/mL of aTc. Plates were grown for 16 hours at 37°C. Cultures were diluted 40 times with Focusing Fluid and directly measured on a flow cytometer.

2-input history-dependent multicellular program characterization

For the 2-input history-dependent program characterization in multi-cell, the history-dependent devices (such as F0, B1, and B2) were co-transformed with the dual controller in J64100 [Bonnet 2013]. For transformation and further culture, media were supplemented with 12.5 μg/mL of kanamycin and 25 μg/mL of carbenicillin.

As a control, the OSiRIS constructs (F0, A2, A3) and a strain without fluorescent protein were used.

For the first day, 96 deep-well plates filled with 500 μL per well of Hi-Def Azure supplemented with glycerol were inoculated with three clones per co-transformation and three clones per control. Plates were grown for 16 hours at 37°C. Cultures were diluted 40 times on Focusing Fluid and directly measured on a flow cytometer according to previously described methods (as for OSiRIS characterization). From the stationary phase culture, cells (B1, B2 co-transformation) were mixed in identical proposition: such as 100 μL for each culture in a 96 deep-well plates. For growth, each mix of cultures was diluted 1, 000 times in Hi-Def Azure-glycerol in three different induction conditions, no inducer, 1% of arabinose, or 200 ng/mL of aTc. Plates were grown for 16 hours at 37°C. Cultures were diluted 4 times in PBS and measured in the plate reader using BioTeck Cytation 3 with the following parameters (GFP: excitation 485 nm, emission 528 nm, gain 80, BFP: excitation 402 nm, emission 457 nm, gain 70, RFP: excitation 555 nm, emission 584 nm, gain 100, Absorbance: 600 nm). For the second induction, each culture from the previous induction was diluted 1, 000 times in fresh HI-Def Azure-Glycerol, with no inducer, 1% of arabinose, or 200 ng/mL of aTc. Plates were grown for 16 hours at 37°C. Cultures were diluted 4 times in PBS and measured in the plate reader with the previous parameters.

For the multicellular experiment, only one experiment was performed. For each sample,

GFP, BFP, and RFP fluorescence intensity over absorbance at 600 nm were calculated and the mean value was calculated between the three replicates. The fold change over the negative control was determined from this mean value over that of the negative control. The error bars correspond to the standard deviation between the different replicates.

Chapter 4

Design of scalable single-cell recombinase logic

Contents

In this chapter, I will present my work on the design of scalable single-cell recombinase logic. The first part of this work is the development of a framework for combinatorial design of single-cell integrase logic based on the generation of a complete set of circuit design. This database is available on a web interface called Recombinator. The second part of this work is the use of the Recombinator database for the design and characterization of all single-cell 3-input logic circuits.

This work was the fruit of a collaboration between Jerome Bonnet and myself and Michel Leclère, Guillaume Kihli and Federico Ulliana from the LIRMM in Montpellier. I was at the origin of the project and of the collaboration with the LIRMM. The Recombinator generation and the web-interface were developped by Michel Leclère and Guillaume Kihli, and the idea and algorithm was conceived through collaborative discussion with Michel Leclère, Federico Ulliana, Jerome Bonnet, Guillaume Kihli, and myself. I was at the origin of the simplification based on P- and NP-class after fruitful discussions with Guillaume Kihli, Michel Leclere, and Jerome Bonnet.

4.1 RECOMBINATOR: a framework for combinatorial design of single-cell integrase logic

4.1.1 Introduction

In the first part of my thesis, I presented design methods for implementing Boolean and sequential logic within multicellular systems. The use of distributed multicellular computation allows the simple implementation of complex logic function through the combination of a reduced number of well-characterized components. However, some applications may require implementation of complete logic function in a single cell. For example, when working in a multicellular organism, computing at the single cell level is crucial so that each cell can respond independently to spatially distributed signal patterns. As another example, therapeutic bacteria operating *in vivo* would freely navigate through the organism and therefore the full computational circuit would have to be implemented in individual cells.

Another reason to pursue single-cell recombinase logic is to push the limitation of logic circuit compaction. First of all, we wanted to know whether it was possible at all to implement all 3 and 4-input logic functions in a single cell using the design strategies previously developed for 2-input logic [Bonnet 2013, Siuti 2013].

In other words, how well do these designs scale? As we will see, we found that while this scheme can implement all 3-input functions, they start to be limited for 4-input ones.

Single-cell logic circuits based on repressors were built in *E. coli* and are the largest (in bp) synthetic circuits implemented to date. The use of repressors requires multilayer implementation, which results in large genetic circuits. This approach requires a large number of orthogonal components that are challenging to obtain and time-consuming to optimize. Moreover, the high number of parts can cause metabolic burden and affect cellular viability.

Integrase-based single-cell logic circuits are an alternative to repressor-based systems. These circuits use the integrase's enzymatic activity to excise or invert DNA sequences. The first example of such designs came from the work of [Bonnet 2013], in which 2-input logic functions were implemented using terminator-based switches (transcriptors), leading to a highly-compact architecture. As an example, a 2-input XOR function is based on a terminator surrounded by the two integrase site pairs. A systematic framework for implementing of up to 3-input logic functions was developed by [Weinberg 2017] in mammalian cells. To do so, a scaffold for implementing all circuits was build based on integrase site variants responding orthogonally to the same integrase. Consequently, Weinberg and colleagues circuits are not as compact as Bonnet and colleagues designs. Moreover, to scale-up this design, it is not clear how many pairs of mutant recombination sites can be used in parallel in single-cell without any non-specific recombination reaction occuring [Colloms 2014]. Finally, repetitive DNA sequences often lead to genetic instability through homologous recombinations, and highly-repetitive DNA sequences

are notoriously difficult to synthesize.

Our objective was thus to design the most compact, robust, and reliable logic circuits. Consequently, we made deliberate choices in the initial design steps: to not use integrase site variants to limit the above mentioned problems and to not use multiple integrases per input to keep circuits within a reduced size range.

Below is the table summarizing the specifications for our logic system and their motivations (Table 4.1).

We were aware that these limitations may prevent the design of some circuits. Yet, by setting these strong constraints, we aimed to push the limit of logic circuit compaction.

Design Specficitions	**Motivation**
Single cell	- Applications requiring long-term usage and targeted field release. - Reduce growth competition problem. - Fundamental question: is it even possible? How far can we go?
Use of serine integrases	- Irreversible, memory, stored within DNA. - Permit compact circuits. - Work in large number of organisms.
One pair of sites/integrase	- Reduce problems of non-specific recombination. - Reduce genetic instability. - Reduce difficulties to synthesize. - Reduce the size of the circuit.
One integrase by input	- Reduce the number of orthogonal integrase needed. - Reduce metabolic load to the cell. - Reduce the size of the circuit.
Regulation of transcription using promoters and terminators	- Simple set of tools. - Two tools for opposite behaviors.

Table 4.1: Motivations for the specification of our logic design.

Exploiting the compactness of integrase-based circuits, we designed single-layer logic circuits. Consequently, no design rules developed for electronic or multi-layer bio-logic circuit could be applied to single-layer, single-cell integrase logic circuits.

Integrase logic circuits were originally designed by hand, in a trial-and-error manner. This strategy worked well for 2-input devices, but was already cumbersome when applied for 3-input devices, even when performed by experts (i.e. ourselves). For instance, we could not find any design to implement some functions. Moreover, even when we found designs implementing a

particular function, we had limited possibilities to optimize the design.

In order to explore the full design space, we thus turned to a combinatorial approach in which we generated millions of combinations and permutations of sites, genes, and regulatory elements. Because the number of generated sequences would still have been enormous and highly redundant, we defined a strategy to reduce the number of generated sequences while conserving the completeness of the logic device set. Indeed, we developed a formal language allowing us to represent logic circuits with the essential information. The integrase site array is represented in a simplified manner as a logic structure. The logic designs incorporating regulatory elements are represented as a logic architecture. In these architectures, the inputs are not attributed to specific sites. Sites are associated to inputs in the web-interface, linking the resulting sequence to a specific Boolean function.

Once generated, this device library could be filtered according to different parameters (e.g. total size, number of genes, promoters, use of inversion or excision), providing us with a much more efficient way to navigate through the recombinase logic design space.

4.1.2 Definition of a formal language to permit the generation of a design database

We implement logic using integrases. An integrase targets specific integrase sites and mediates DNA excision or inversion between its specific integrase sites according to their relative orientations. To obtain a complete and reduced generation, we generated logic structures corresponding to integrase site arrays but with the smallest possible amount of information (Figure 4.1).

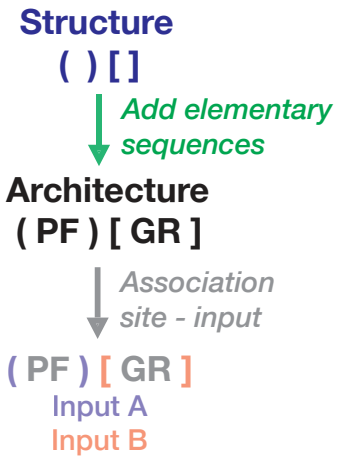


Figure 4.1: **General workflow of the generation.**

4.1.2.1 Logic structures

We use the term "logic structure" for a string representing a particular combination of two or more pairs of recombination sites targeted by different enzymes. A logic structure contains the following information: (i) the number of different pairs of sites, (ii) their relative positions, and (iii) their mode of recombination: inversion or excision.

As in our Boolean logic implementation, integrase sites do not interdigit; they can be associated to a nested sequence of brackets, corresponding to the Dyck language [Autebert 1987]. As integrase site pairs can react in two manners, a pair of sites is represented by parentheses ( ) when being inverted or by brackets [ ] when being excised. For example, we can have a logic structure responding to 2 inputs: ( ) [ ], or 3 inputs: (( ) [ ]). This notation can unambiguously represent all instances of non-interlaced recombination sites through so-called well-balanced parentheses.

We also wanted to represent sites once recombination has happened ("used sites"). Inverted sites were represented by {}and excised sites by |. For example, the structure ( ) [ ] can have three states upon recombination: {}[ ] , ( ) |, or {}|.

For the generation of logic structure, Dyck words with the corresponding number of inputs are generated and then functionalized to either excision [ ] or inversion ( ) to obtain all possible combinations.

4.1.2.2 Logic architectures

We then generated logic architectures by inserting genes, promoters, and terminators at different places within the logic structure. In order to do so, we had to define symbols representing the different components. Each symbol was attributed a semantic describing its function. The semantic contains two pieces of informations: (i) the function itself (transcription, termination, etc) and (ii) the direction on the linear DNA sequence in which the function is operational (forward or reverse). We then defined rules to combine neighboring semantics. Using this framework, we could infer the gene expression status of any concatenation of promoters, terminators, and genes. These rules are defined in subsection 3.

To obtain the most compact designs, we defined a set of elementary sequences corresponding to irreducible semantic compositions. Logic architectures are then generated by placing elementary sequences between brackets within logic structures. From a logic architecture, derived architectures are obtained by simulating all possible recombination events. Sequences flanked by excision brackets [ ] are removed, while sequences flanked by inversion brackets ( ) are flipped (Figure 4.2). The semantic of each derived architecture is then obtained based on the rules defined in the following subsection. Inputs are then attributed to particular brackets to identify the corresponding Boolean functions.

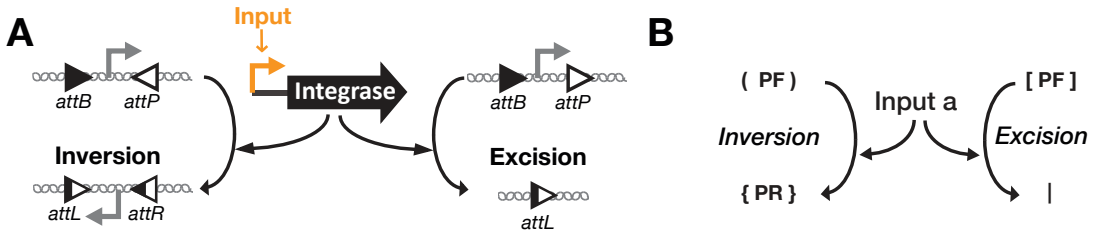


Figure 4.2: **A - Integrase action based on diagram, B - Correspondence to logic structures.**

4.1.3 Ontology of synthetic gene circuits

4.1.3.1 Transcriptional components and their semantics

Here I will set forth the rules used to determine the output state of any sequence.

We used three elementary types of parts involved in transcription: (1) promoters and (2) terminators respectively initiate and terminate the flow of RNA polymerase; and (3) parts which are transcribed (e.g. genes) (Table 4.2).

As the RNA polymerase flow is oriented, a promoter initiates transcription in only one orientation and is inactive in the other. A terminator can block transcription in either a single orientation or both orientations. For simplification, we considered here that the terminators used are asymmetric and therefore terminate transcription in a single orientation.

In natural systems, the parts transcribed are usually sequences controlling translation (e.g. RBS or Kozak) with the CDS (coding sequence) for expression of protein, or a non-coding RNA only. In our system, we focused on protein expression as output and called "gene" the concatenation of the 5' UTR sequence controlling translation (e.g. RBS or Kozak) with the CDS (coding sequence) and the 3' UTR sequence to terminate transcription (terminator).

We positioned these parts in different orientations to control the transcription of the output gene. Of note, we considered sequences containing several copies of the same gene (encoding for the same protein), but not containing genes encoding for different proteins.

For each transcriptional part, we defined a semantic, corresponding to the part activity related to transcription (the "meaning" or function of the biological part), and an associated symbol for simplification of further explanations. The symbol corresponding to each part is listed in Table 4.2. The semantics are: promotion of transcription (fP), termination of transcription (fT), encoding of a gene (fG) and no activity (fN). Each part encodes two distinguishable semantics, one in each orientation (forward or reverse).

In this algorithm, we simplified part activity related to gene expression; transcription mechanism is considered as a binary, digital process in which a gene is either ON or OFF. Obviously, this is an oversimplification of biological systems eluding intermediary activity levels which

Transcriptional part and its orientation	Part symbol	Part semantic forward	Part semantic reverse
Promoter in forward	PF	fPF	fNF
Promoter in reverse	PR	fNF	fPR
Terminator in forward	TF	fTF	fNR
Terminator in reverse	TR	fNF	fTR
Gene in forward	GF	fGF	fNR
Gene in reverse	GR	fNF	fGR
Neutral part	N	fNF	fNR

Table 4.2: Definition of the forward and reverse semantic for each transcriptional part.

are relevant in natural biology and which can cause unwanted effects in engineered biological systems. For example, promoters and terminators can have different levels of transcription initiation and termination. Nevertheless, this simplification is essential as our objective is to use these natural mechanisms to implement logic circuits with only two possible states.

4.1.3.2 Ten rules for determining the semantic of transcriptional parts assemblies

Transcriptional parts are concatenated to form transcriptional sequences. We defined a set of rules to determine the semantics of sequences. As any forward and reverse semantics can be considered separately, the following properties are defined considering a single orientation of the construct. For simplification, the properties are written in the forward orientation, from 5' to 3'.

The semantic of a transcriptional sequence corresponds to the concatenation of the semantics of each part of the sequence. Indeed, to determine the semantics of a concatenation of transcriptional parts, we use a step-wise iterative process in which semantics are composed two by two.

This concatenation can be simplified with the following rules in a reduced set of six elementary semantics. These six semantics correspond to the four semantics described previously (fP, fT, fG, and fN) plus the semantics corresponding to the expression of a gene: fX and the composition of fG followed by fP: fGP. As the two-by-two concatenation of the four basic semantics leads to one of these six semantics, this set of semantics is complete (detailed below).

Rules:

(1) Non-commutativity: Concatenation of semantics is not commutative as parts concatenated in a different order does not lead to the same semantic. As an example, PF-GF permits expression of the gene, therefore encoding the semantic fX, which is not the case for GF-PF.

(2) Neutrality: A sequence without any activity in a particular orientation does not affect other sequences placed in the same orientation (i.e. fN is neutral to other semantics similarly oriented).

(3) Assimilation of fX: The semantic of gene expression, fX, assimilates all other semantics. In others words, the composition of the fX semantic with another semantic is simplifiable to fX. In this work, we aim at defining if a construct leads to expression of a gene or not and the composition of fX with another semantic does not affect the fX semantic.

(4) Idempotent: all semantics are idempotent (an operation has the same effect even if applied multiple times), as we consider that the concatenation of two similar parts is equivalent to a single part.

Others rules are due to the mechanism of gene expression. A gene is expressed if it is transcribed by RNA polymerase; consequently, a promoter needs to be positioned upstream without a terminator positioned between the promoter and the gene. This mechanism can be assimilated to a flow that is opened by the promoter, stopped by the terminator, and the system is ON when the flow is at a specific location: the gene.

(5) Expression occurs only if a promoter is placed upstream of a gene without a terminator in between. Such as, the concatenation of the semantic promotion with semantic gene is simplifiable to fX. i.e. fP-fG=fX

(6) A gene followed by a promoter leads to the semantic fGP, as the promoter can be active for a downstream gene and the gene can be expressed by an upstream promoter. Consequently, **fG-fP=fGP**. In this case, we have associativity of the semantics fG and fP.

(7) If a promoter is followed by a terminator, the RNA polymerase flux is blocked by the terminator, consequently, *fP-fT=fT*.

(8) If a terminator is followed by a promoter, the terminator will have no effect on the semantic of the sequence as the terminated transcription will be re-initiated by the promoter, consequently: **fT-fP=fP**.

(9) If a terminator is followed by a gene, the gene cannot be expressed as the RNA polymerase flux is blocked by the terminator; consequently, **fT-fG=fT**.

(10) If a gene is followed by a terminator, the terminator will have no effect on the semantic of the word; indeed, if the previous semantic is fP, it will result in the expression of the gene, consequently: **fG-fT=fG**.

Mathematical definition of the rules for semantic composition. For two semantics fA and fB, fA-fB is the concatenation of fA with fB, fA being in 5' and fB in 3'.

(1) Not commutative: fA-fB≠fB-fA

(2) Neutrality of fN: fN-fA=fA and fA-fN=fA

(3) Assimilation by fX: fX-fA=fX and fA-fX=fX

(4) Isomorphisme: fA-fA=fA

(5) Condition of gene expression: fP-fA-fG=fX only if fA≠fT ou fA=fP

(6) Composition of Gene-Promoter: fG-fP=fGP

(7) A terminator cancels promotion: fP-fT=fT

(8) A promoter cancels termination: fT-fP=fP

(9) A terminator block transcription of a following gene: fT-fG=fT

(10) A terminator cannot block transcription of a previous gene: fG-fT=fG

We concatenated the six previously defined semantics two by two. Using the previously defined rules, all concatenations of these six semantics are simplifiable to one of the six semantics (Table 4.3). Therefore, the set of semantics is complete and our rules are scalable to the concatenation of N transcriptional parts.

5' to 3'	**fN**	**fP**	**fT**	**fG**	**fGP**	**fX**
fN	fN	fP	fT	fG	fGP	fX
fP	fP	fP	fT	fX	fX	fX
fT	fT	fP	fT	fT	fP	fX
fG	fG	fGP	fG	fG	fGP	fX
fGP	fGP	fGP	fG	fX	fX	fX
fX	fX	fX	fX	fX	fX	fX

Table 4.3: Simplification of all possible concatenation of the six semantics two by two, as the row corresponding to the semantic in 5' and the column the semantic in 3'.

4.1.3.3 Selection of a set of 26 elementary sequences encoding the 26 possible semantics in a minimized manner

As all sequences have a semantic in forward and in reverse orientations, considering both orientations, 26 semantics exist, as 5 times 5 for the set (fN, fP, fT, fG, fGP) and the semantic expression that assimilates in both orientations.

As our objective was to have the simplest constructions, we aimed at implementing each semantic with the minimal number of parts. We then defined for each semantic one elementary sequence implementing the semantic (Figure 4.3C, Table 4.4).

For sequences with the same number of parts, we selected sequences to optimize the biological implementation according to two criteria derived from experimentally validated biological constructs. First, we avoided promoters facing each others, as two RNA polymerase flows might interact and create unexpected behavior (Figure 4.3A) [Boque-Sastre 2015] [Uesaka 2014]. Sec-

ond, we reduced the number of parts between a gene and the promoter initiating its transcription (Figure4.3B). Indeed, RNA polymerase tends to unbind from DNA, and transcription efficiency decreases with the increase of distance between the gene and the promoter [Chizzolini 2014]. For some semantics, several sequences are equivalent to each other,where upon one is chosen arbitrarily, such as for TR-TF, the equivalent of TF-TR (for the semantic: fTR/fTF).

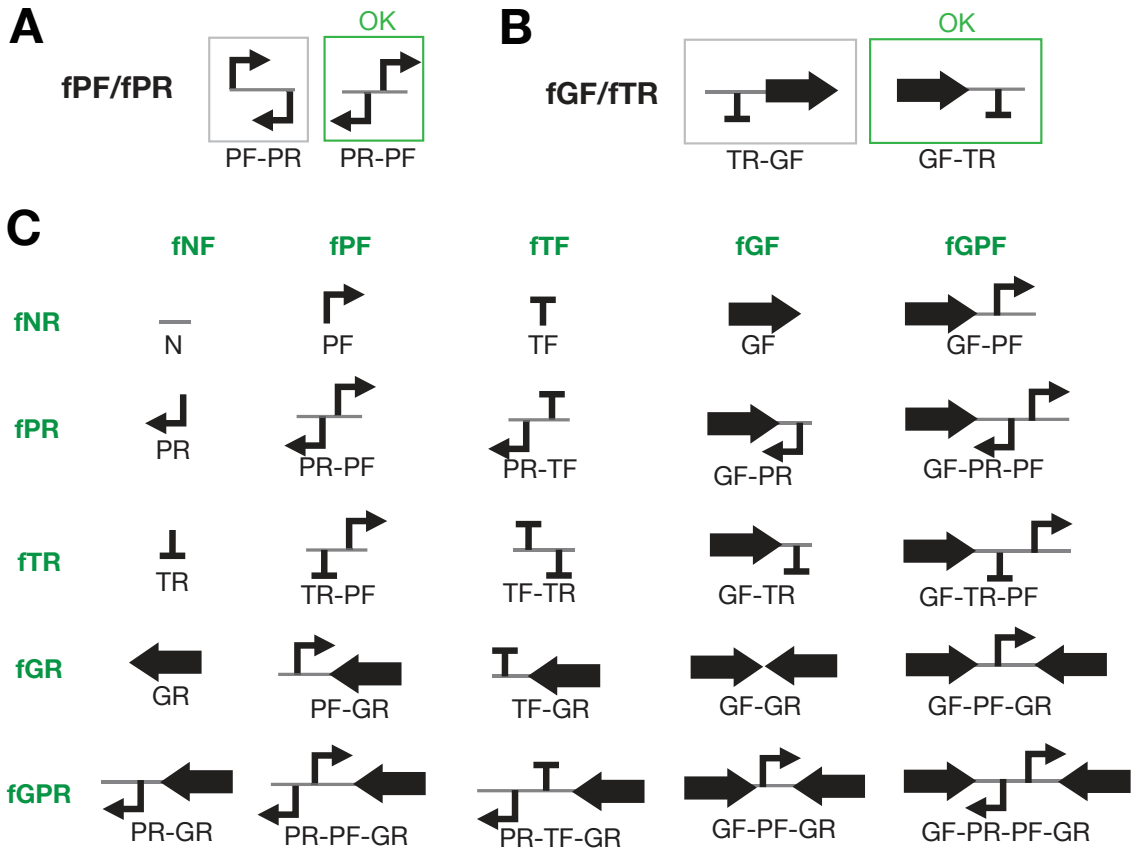


Figure 4.3: **Selection of the 26 elementary sequences.** A - Selection of an elementary sequence by avoiding promoters facing each other. Both sequences implement the semantic: fPF/fPR. PR-PF is preferred to PF-PR. B - Selection of an elementary sequence by limiting the space between a promoter and its transcribed gene. For the semantic fGF/fTR, GF-TR is chosen instead of TR-GF, for if PF is placed upstream, the space between PF and GF is larger for the TR-GF sequence. C - Elementary sequences and diagrams for each combination of forward and reverse semantics.

The semantic fX is not represented in this table as fX assimilates other semantic. fX can be implemented by either PF-GF or GR-PR. These 26 elementary sequences correspond to the domain of sequences placed between sites during the generation (Figure 4.3C).

4.1.4 Generation of all possible sequences

Our objective was to generate all possible irreducible logic architectures based on integrase Boolean logic from 1 to 4 inputs.

Semantics	fNF	fPF	fTF	fGF	fGPF
fNR	-	PF	TF	GF	GF-PF
fPR	PR	PR-PF	PR-TF	GF-PR	GF-PR-PF
fTR	TR	TR-PF	TR-TF	GF-TR	GF-TR-PF
fGR	GR	PF-GR	TF-GR	GF-GR	GF-PF-GR
fGPR	PR-GR	PR-PF-GR	PR-TF-GR	GF-PR-GR	GF-PR-PF-GR

Table 4.4: The set of elementary sequences for implementation of each semantic. The first column corresponds to the semantic in reverse orientation and the first row semantic in forward orientation. The elementary sequences selected to implement the corresponding forward and reverse semantics are represented here.

An irreducible logic architecture is an architecture with the following characteristics: (1) the sequences placed between the sites are elementary sequences, and (2) each elementary sequence is functional in at least one derived sequence. In other words, the elementary sequences cannot be replaced by any other elementary sequence without changing the semantic of the architecture or of a derived architecture (hence without changing the corresponding logic function).

As explained before, to generate architectures we first generated logic structures (sequences of parentheses) and then filled each space between parentheses with elementary sequences (from the list of 26). To avoid generation of reducible architectures and generation of inverted architectures, several constraints were used to reduce the set of elementary sequences possible that could be incorporated at a specific locus. The irreducibility of an architecture was then verified by generating all derived architectures. Irreducible architectures are saved in the database and associated to a class of Boolean functions, corresponding to a permutation-class (P-class). This is because a given architecture can implement all members of a P-class by interchanging the input/site attributions.

4.1.4.1 Generation of logic structures

For the generation of logic structures, all the Dyck words of size 2N (where N is the number of inputs) are generated and then functionalized (composition of parentheses and/or brackets). To do so, a construction by induction was used [Manber 1988].

A sequence is equivalent to its reverse complement; called here inverse. To reduce the number of architectures, we avoided generation of an architecture and its inverse. To do so, we acted on several steps of the generation process, such as the construction of the logic structures. Indeed, some logic structures are inverses of one another, e.g.(( )) ( ) is the inverse of ( ) (( )). Two inverse structures will then lead to the generation of inverse architectures; thus, only one structure was conserved for the next step of the generation. All inverse sequences are not removed by removing inverse structures. Indeed, some structures are palindrome, e.g. (

( )). Architectures based on palindromes will either be a palindrome or will have an inverse generated from the same structures. Elimination of these inverses is explained later on.

4.1.4.2 Constraints on domain of sequences for each variable

Each space between integrase sites, such as parentheses or brackets, is considered as a variable. During the generation process these spaces are filled with elementary sequences from a specific domain. A domain is a set of elementary sequences that can be placed at a defined position during the generation (e.g. the beginning of the sequence or between sites in excision). To reduce the generation of reducible architectures, constraints are defined to minimize the domain of elementary sequences at each position.

First, at the extremity of architectures, many elementary sequences are useless. For example, PR at the beginning of an architecture is useless as it cannot lead to gene expression. We thus defined a reduced domain of elementary sequences for the left (5') side of the architectures (PF, GF, PF-GR, N) and for right (3') side(PR, GR, GF-PR, N).

Then, the semantic fX (expression) cannot exist outside of an excision module; otherwise, architectures can be reduced as a True function composed only of fX. This semantic is then only present in domains for variables which are inside an excision.

For atomic excision and inversion (excision and inversion without integrase sites in between), some semantics are always useless as the action of the integrase will not change the semantic of the sequence. For instance, for atomic inversion, all semantics which are invariant with inversion are useless. As such the sequences N, PR-PF, TF-TR, GF-GR, and GF-PR-PF-GF are useless in atomic inversion. After excision, as the sequence is deleted, the semantic between sites is fN, so that only the fN semantic is useless in an atomic excision.

Finally, to avoid the generation of inverse sequences from palindromic Dyck words, we authorized some semantics in a specific variable according to the semantic of the previous variable. Before the generation, a program creates a list of authorized patterns restricting the generation of sequences and their inverses.

4.1.4.3 Algorithm CSPs (Constraint Satisfaction Problems)

For the generation process, a constraint satisfaction problem algorithm is used [Kumar 1992]. The variables are the sequences between the sites and the domain of variables are the elementary sequences that can be affected to these variables. The constraints are those detailed previously.

During the execution of the algorithm, one value of the domain is assigned to a variable and the constraints are verified. If they are not respected, another value is assigned. Otherwise, a value is assigned to the next variable. The assignment is performed from the right to the left of the structure. When all variables are affected, a full architecture is obtained. Its irreducibility

and non-simplifiability to a Boolean function with a lower number of input is verified by generation of the derived architectures. Then, the generator passes to the following value in the domain of the last affected variable. The algorithm stops when all the domains of each variable have been browsed.

4.1.4.4 Generation of all architectures for up to four inputs

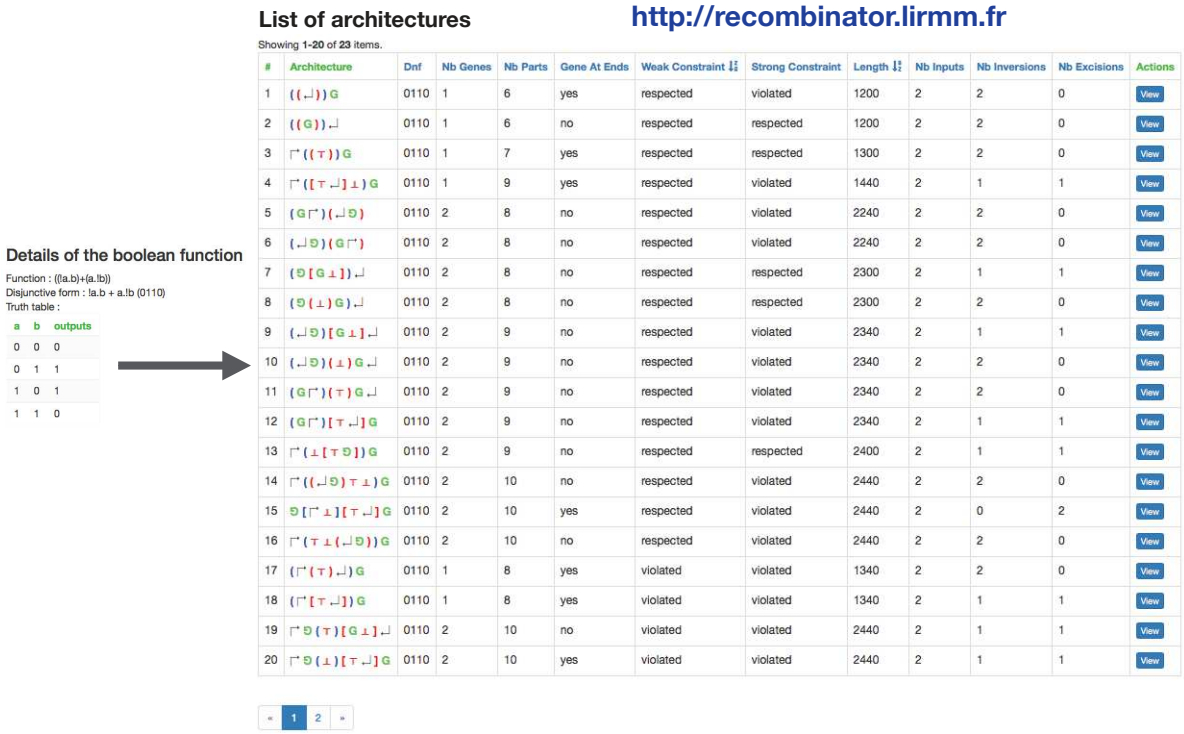
We generated all architectures for up to four inputs, for a total of 96,965 architectures for three inputs and around 18,668,046 architectures for four inputs. In our generation, we obtained for the implementation of each 3-input Boolean functions several architectures. Consequently, all 3-input Boolean functions are implementable in single cell using one integrase per input and a single integrase site pair per integrase. For four inputs, we obtained architectures for the implementation of 91% of the 4-input Boolean functions. Thus, all 4-input Boolean functions are not implementable in single cell using one integrase per input and a single integrase site-pair per integrase. We have not done the generation for an increasing number of inputs as it required large storage capacity (1.2Gb for 4-inputs), but we can suppose that the percentage of implementable functions will decrease with the increase of the number of inputs.

4.1.5 A web-interface for exploring on the database

Using this generation process, we obtained 96,965 architectures for three inputs and 18,668,046 for four inputs. Each 3-input Boolean function is implementable with several architectures. For 3-input Boolean function, from 20 to 5833 different architectures permit the implementation of the same Boolean function (for a mean of 135 architectures/function). Therefore, to find architectures corresponding to a specific Boolean function, we developed a web-interface called Recombinator.

In the Recombinator web interface, the user provides as an entry her (his) Boolean function of interest, and the web interface shows the architectures permitting the implementation of this Boolean function. By selecting one architecture, the architecture is shown with the association of the sites to the input of the corresponding Boolean function. Additionally, all logic functions implementable with the same architecture (therefore P-equivalent functions) are accessible from this page (Figure 4.4).

The web interface also allows the users to sort architectures according to various biological criteria for implementation, such as the size of the architecture, the number of genes, the number of parts, if the gene is at the end, and if promoters faces each others or not. We selected an elementary sequence for each semantic according to previously described constraints, but as the architecture is composed of various elementary sequences these constraints are not especially respected in the generated architectures. This option to sort architectures will help choosing which architecture to implement.



#	Architecture	Dnf	Nb Genes	Nb Parts	Gene At Ends	Weak Constraint	Strong Constraint	Length	Nb Inputs	Nb Inversions	Nb Excisions	Actions
1	((↵))G	0110	1	6	yes	respected	violated	1200	2	2	0	View
2	((G))↵	0110	1	6	no	respected	respected	1200	2	2	0	View
3	⌐((⊤))G	0110	1	7	yes	respected	respected	1300	2	2	0	View
4	⌐([⊤↵]⊥)G	0110	1	9	yes	respected	violated	1440	2	1	1	View
5	(G⌐)(↵Ð)	0110	2	8	no	respected	violated	2240	2	2	0	View
6	(↵Ð)(G⌐)	0110	2	8	no	respected	violated	2240	2	2	0	View
7	(Ð[G⊥])↵	0110	2	8	no	respected	respected	2300	2	1	1	View
8	(Ð(⊥)G)↵	0110	2	8	no	respected	respected	2300	2	2	0	View
9	(↵Ð)[G⊥]↵	0110	2	9	no	respected	violated	2340	2	1	1	View
10	(↵Ð)(⊥)G↵	0110	2	9	no	respected	violated	2340	2	2	0	View
11	(G⌐)(⊤)G↵	0110	2	9	no	respected	violated	2340	2	2	0	View
12	(G⌐)[⊤↵]G	0110	2	9	no	respected	violated	2340	2	1	1	View
13	⌐(⊥[⊤Ð])G	0110	2	9	no	respected	respected	2400	2	1	1	View
14	⌐((↵Ð)⊤⊥)G	0110	2	10	no	respected	violated	2440	2	2	0	View
15	Ð[⌐⊥][⊤↵]G	0110	2	10	yes	respected	violated	2440	2	0	2	View
16	⌐(⊤⊥(↵Ð))G	0110	2	10	no	respected	violated	2440	2	2	0	View
17	(⌐(⊤)↵)G	0110	1	8	yes	violated	violated	1340	2	2	0	View
18	(⌐[⊤↵])G	0110	1	8	yes	violated	violated	1340	2	1	1	View
19	⌐Ð(⊤)[G⊥]↵	0110	2	10	no	violated	violated	2440	2	1	1	View
20	⌐Ð(⊥)[⊤↵]G	0110	2	10	yes	violated	violated	2440	2	1	1	View

« 1 2 »

Figure 4.4: **Example of a search on the Recombinator web-interface.** With the XOR logic function as input, 23 architectures are found, with the 20 first architectures shown here. This list can be sorted according to various criteria.

The web interface is still under construction, and while most of the functions are already accessible, the descriptions on how to use the interface are lacking. The current version can be found at `http://recombinator.lirmm.fr`.

4.1.6 Discussion

Through an exhaustive generation of integrase-based architectures, we proved that all 3-input and 91% of 4-input Boolean functions are implementable using integrase-based devices in single cell with one integrase and one integrase site pair per input.

We did not generate all possible architectures; rather, we generated only irreducible architectures corresponding to architectures with a minimum number of parts and all parts used (functional) in at least one derived sequence. We believe that the best designs are the simplest one.

As we developed an algorithm that generates only non-equivalent, irreducible architectures, we succeeded generating architectures for 4-input using reduced computational resources. We avoided the generation of mirror architectures and of reducible architectures by defining a set of rules. However, this brute-force generation is still limited to four inputs. Nevertheless, this exhaustive database could be used to define rules for the systematic design of devices implementing Boolean functions of increasing number of inputs. One option would be to compose

4-, 3-, and 2-input devices to obtain 5-input devices.

Each 3-input Boolean function is implementable with a large set of architectures, and each architecture is implementable by many different DNA sequences. These architectures theoretically permit the implementation of these Boolean functions, but it is clear that many architectures will not behave as predicted when implemented experimentally. We aimed at selecting architectures that will most likely permit robust implementation in living organisms. To do so, we provide the option to sort architectures according to several criteria, which could be used to select the less expensive sequences (usually the shortest ones) and the sequences which have the highest probability to be functional according to our knowledge on circuit implementation.

This database and web interface would be useful for all biological logic designers for the design of minimized integrase-based single-cell logic circuits.

We also described rules for the definition of semantics and semantics combination that could be applied to determine the gene expression status of any given construct incorporating those elements. This framework could be extended to incorporate other regulatory elements not described here.

4.2 Using the Recombinator database for the systematic design and construction of all single-cell 3-input logic gates

By generating all possible combinations of integrase targets, we obtained biological designs for implementing all 3-input logic functions and 91% of all possible 4-input logic functions. As the design generation has been performed without biologically informed constraints, we expected that some theoretical designs would not behave as predicted or would require numerous cycles of optimization to do so. By generating this database, we proved that all 3-input logic functions are theoretically implementable in single cell. The remaining challenge now is to show that they are also experimentally implementable.

For each 3-input Boolean function, we obtained an important number of different designs, ranging from 20 to 5,833, depending on the logic function. These designs are theoretical and presented in an abstract form; they represent the general architecture in which a particular combination of integrase sites and regulatory elements can execute a given function. Details like site orientations, sites identities (attB or attP), and their relative positions are not specified, neither is the identity of parts for gene expression.

From each theoretical design, an infinite number of DNA sequences is possible. Consequently, the number of possible DNA sequences able to implement each Boolean function is enormous. For instance, for the 218 Boolean functions strictly responding to 3 inputs, 96,965 logic architectures were generated. For each of these architectures, it seems impossible to test all possible biological implementations. How do we thus validate our designs? Can we defines

rules that could help to filter the most "promising" sequences for further testing? Can we find a method to reduce the number of devices to characterize?

Classes of Boolean functions have been defined, such as P-class for permutation and NP-class for negation and permutation [Jaakko T. Astola 2006]. Based on this classification, we propose to reduce the experimental optimization of logic devices to the implementation of a single Boolean function per NP-class, such as 16 for 3 inputs. By decreasing the number of Boolean functions to implement by 93%, this simplification would allow the precise characterization and optimization of a reduced number of devices. These optimized biological designs could therefore be used to implement all 3-input Boolean functions in single-cell. I did not have time during my Ph.D. to perform these characterizations nor to finalize the selection of the 16 constructions as the Recombinator web-interface has only recently been completed. However, I present below the theoretical foundation for such work.

4.2.1 P-class and its *in vivo* correspondence

4.2.1.1 Definition of P-class

A P-class is composed of all Boolean functions that are P-equivalents [Jaakko T. Astola 2006]. Two functions are P-equivalent only if they can be reduced to one another by permutation of input variables (P) $x_i \leftrightarrow x_j$.

In other words, functions belong to the same P-class if one function can be transformed into another by permuting the inputs. For example, the functions f_1=not(A).B and f_2=A.not(B) are P-equivalent, indeed, if we permute A and B in f_1, f_1^P=not(B).A=f_2 ([Friedman 1986]).

For 2 inputs, P-classes contain 1 or 2 Boolean functions, as one permutation exits (permutation of A to B). Indeed, for the AND gate, the permutation of A to B leads to the same function, and for the example above, not(A).B leads by permutation to A.not(B), leading to a 2 function P-class.

For 3 inputs, P-classes contain 1, 3 or 6 Boolean functions. If all permutations lead to the same function, the P-class is composed of one function, e.g. the function: A.B.C. Then, if only permutation of all inputs leads to different functions, the P-class is composed of three functions, e.g. the functions: A.B+C, A.C+B and B.C+A. Finally if any permutation leads to different functions, the P-class is composed of 6 functions.

The number of permutations with N inputs is N!

4.2.1.2 Permuting logic device inputs by switching connection between integrase and inducible promoters

P-equivalence corresponds to an equivalence of function by permutation between input variables. In integrase-based systems, logic implementation is decoupled from the connection to input signals. The logic function is implemented by computational devices composed of integrase sites and parts for gene expression and input control via the expression of integrases.

Consequently in our biological circuits, we can easily permute inputs by changing the connection between integrases and inducible promoters responding to the different inputs. Then, all Boolean logic functions from one P-class are implementable using the same computation device. Indeed, we used this property in Chapter 2, Section 1 to reduce the number of logic devices to characterize for multicellular Boolean logic implementation. For example, for the implementation of not(A).B and not(B).A, the same computational device is used and connections between integrases and inducible promoters are inverted (Figure 4.5).

Therefore, the characterization and optimization of one logic device per P-class is sufficient for implementing all logic functions of this P-class. Consequently, the number of logic devices to characterize decreases by 68% (from 218 to 69) for 3 inputs (Table 4.5).

The generation of logic devices with Recombinator was programmed to generate architecture without specification of integrase site identity, taking the decoupling of inputs / logic devices into account and therefore reducing the size of the generation.

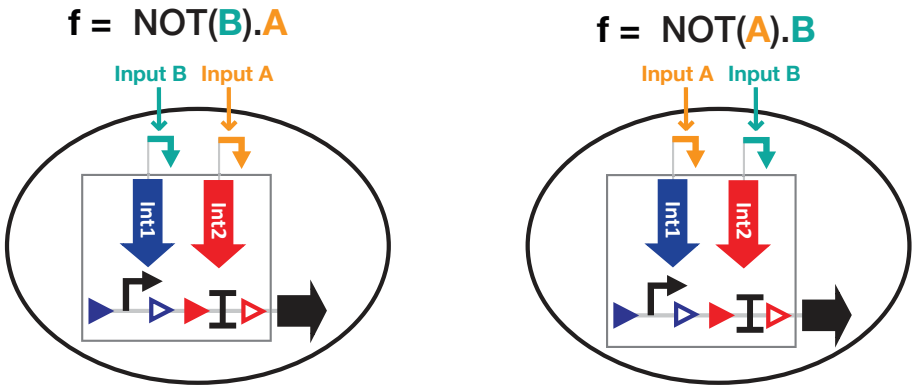


Figure 4.5: **Implementation of P-equivalent Boolean function using the same logic devices and different inducible promoter-integrase connections.** Example of the implementation of the P-equivalent NOT(B).A and NOT(A).B logic functions using one logic device and by switching the connection of inducible promoters and integrases.

4.2.2 NP-class and its *in vivo* correspondence using DNA inversion

4.2.2.1 NP-class definition

A NP-class is composed of all Boolean functions that are NP-equivalents. Two functions are NP-equivalents only if they can be reduced to one another by negation of input variables (N) $x_i \leftrightarrow$ not(x_i) and by permutation of input variables (P) $x_i \leftrightarrow$ xj. e.g. f_1=not(A).B and f_2=A.B, f_1 and f_2 are NP-equivalent as by A $\leftrightarrow$ not(A), f_1=f_2.

P-equivalent functions are NP-equivalents.

16 NP-classes exist for strictly 3-input Boolean functions and 380 for strictly 4-input Boolean functions (Table 4.5).

	1 input	**2 inputs**	**3 inputs**	**4 inputs**	**5 inputs**
# functions with strictly N inputs	2	12	218	64,594	4.3 10 9
# P-classes with strictly N inputs	2	8	68	3904	3.7 10 7
# NP-classes	1	5	16	380	1,227,756

Table 4.5: Number of functions, P-classes, NP-classes for a given number of inputs.

4.2.2.2 Negation equivalence using inversion

Here, the objective is to find a correspondence between integrase-based logic implementation and NP-equivalence of functions. As seen above, permutation of variables is performed by changing integrase/input connection. Now, we have to find a correspondence between negation of variables and integrase-based implementation.

Bonnet and colleagues implemented 2-input logic gates based on simple terminator switches. In this paper, NOT and IDENTITY functions are performed using an asymmetric terminator surrounded by integrase sites in opposite orientation for inversion (Figure 4.6A). For the NOT function, the terminator is placed in the OFF orientation, then in absence of input, the output gene is ON. In presence of the input, the terminator is inverted and transcription of the output gene is blocked. For the IDENTITY function, the terminator is placed in ON orientation, then the transcription is possible only in presence of the input, when the terminator is inverted. In this example, the design of NOT and IDENTITY functions differ only by the orientation of the part placed between inversion sites, here the terminator. By inversion of the element between the integrase sites of the IDENTITY device, the negation of the input is performed as we obtained the design of the NOT function except for the integrase sites.

As these designs are based on DNA inversion, the DNA state of the IDENTITY device in the presence of the input corresponds to the negation function, i.e. to the NOT devices. This is

also true to switch from the NOT to the IDENTITY device. When using excision to implement IDENTITY and NOT devices as in Chapter 2, the previous property is not true anymore as excision is destructive.

The previous one-input example shows that the negation of inputs in a logic device based on inversion can be performed by inversion of the element between the corresponding integrase sites. This property is scalable.

As example for the 2-input AND logic device based on DNA inversion, the device is based on a promoter in reverse orientation surrounded by integrase sites placed in series with an asymmetric terminator surrounded by integrase sites (Figure 4.6B). The two inputs have to be present to permit expression of the output gene, so the terminator will not block the RNA polymerase and the promoter will be in the correct orientation for output gene expression. By inverting the orientation of the promoter and terminator between integrase sites, different logic functions are implemented. Indeed, the inversion of the promoter permits the negation of one input, therefore implementing the NOT(A).B logic function. The inversions of both the terminator and the promoter negate two inputs, and therefore permit implemention of the NOT(A).NOT(B) logic function. These logic functions correspond to a 2-input NP-class, and we pass from one function device to another by inversion of the element between the integrase sites corresponding to the negation of the input(s). Moreover, as for the previous one input function, each logic device design (detailed in Figure 4.6C) corresponds to the DNA states of one logic device in a different input state, except for the integrase sites.

To summarize, the negation of inputs in a logic device is performed by inversion of the element between the integrase sites responding to this input. This is true only for devices using DNA inversion. The permutation of inputs in integrase-based circuits is performed by permutation of the connection between integrases and inducible promoters.

Consequently, excepted for integrase sites, the different DNA states of one logic device correspond to the devices for the implementation of a complete NP-class. Therefore, we approximated that the characterisation of one construct per NP-class based on DNA inversion is sufficient to prove the feasibility of implementing all logic functions of this NP-class using this logic device architecture. As 16 3-input NP-classes exist, we reduced the number of Boolean functions to optimize from 218 to 16, a 93% decrease.

4.2.3 Using the Recombinator database to select inversion-based logic devices

Using our Recombinator database, we found that all 3-input Boolean functions are implementable using exclusively inversion-based logic devices. Therefore, it is possible to select, characterize, and optimize one inversion-based logic device per 3-input NP-class.

To do so, we can select any 3-input Boolean function from each NP-class and search for

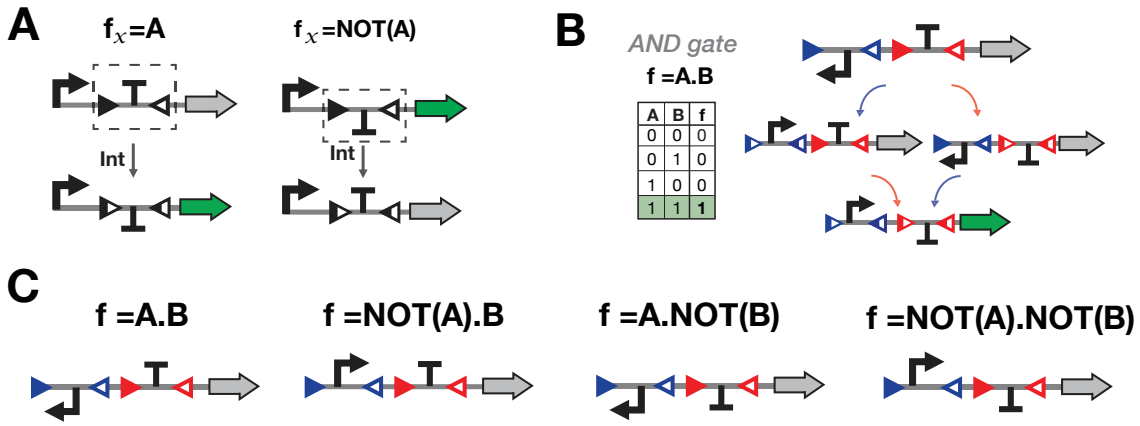


Figure 4.6: **Implementation of NP-equivalent Boolean functions using inversion-based logic devices.** (A) IDENTITY- and NOT- logic devices based on the inversion of an asymmetric terminator. (B) AND gate based on the combination of promoter and terminator modules and its intermediate recombination states. (C) Implementation of all functions from the AND gate NP-class. Designs are based on the AND gate design presented in B and corresponds to the intermediate recombination states of the AND gate, except for the integrase site identity.

implementation in the database. From the set of inversion-based architectures implementing the selected Boolean function, we have to select one or few architectures that will permit implementation of the desired Boolean function with the reduced number of optimization cycles.

Criteria for the selection of architectures are not absolute and depend mainly on the chassis organism, on the available genetic parts, and on the sensibility of the designers. For us, several criteria are important:

1 - The space and complexity between the promoter and the transcribed output gene should be minimized in each DNA state. Indeed, the transcription can be compromised if the space between the promoter and the gene to be transcribed is thousands of base pairs and if the promoter face another promoter in reverse orientation.

2 - The size of the construction to minimize the synthesis cost or cloning complexity. This size corresponds mainly to the number of output genes in the construct.

Using the sorting function of the Recombinator web-interface, we selected 16 logic devices, one per 3-input NP-class.

Using the previous web-interface lacking the sorting functions, we selected, without scanning all possible architectures, a set of 16 architectures for the implementation of the 16 3-input NP-classes (Figure 4.7). This subset of architectures provides an idea of the complexity of the implementation of some NP-classes, but they probably do not correspond to the optimum architectures as they have been chosen manually.

For 4 inputs, 91% of Boolean functions are implementable in single-cell, and probably a similar proportion of NP-class. Therefore, the same strategy can be applied for the implemen-

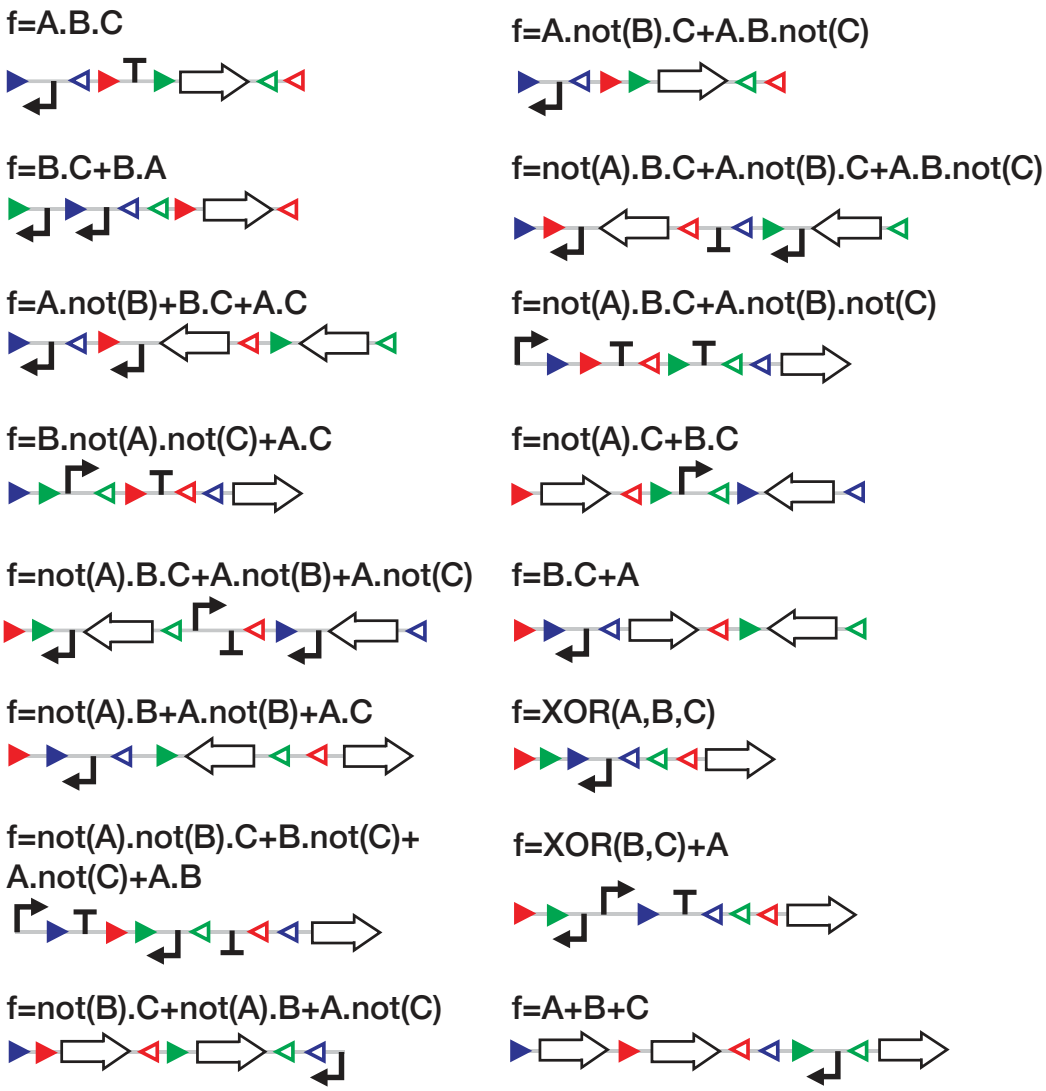


Figure 4.7: Example of possible designs-these are probably non-optimal

tation of most 4-input Boolean functions in single-cell. However, for now, we do not have the proportion of Boolean functions (or NP-classes) which are implementable using inversion only. Some bugs are still present in the web interface for representation of 4-input architectures. With the final web interface, we will determine the number of NP-class implementable using inversion only.

4.2.4 Discussion

We made a parallel between the mathematically defined P-equivalence and NP-equivalence of Boolean functions and the permutation and negation of inputs in integrase-based logic systems. Two P-equivalent Boolean functions are implementable using the same logic device and different integrase-inducible promoter connections. For inversion-based devices, the negation of one input in an integrase-based logic system is performed by inversion of the element between the

corresponding integrase site pair. Two integrase-based logic systems implementing two NP-equivalent Boolean functions can be reduced to each other by switching integrase-inducible promoter connections and by inversion of elements between integrase-site pairs.

Consequently, we propose to characterize and optimize a single inversion-based logic device per NP-class. The DNA states of this logic device correspond to the logic devices for the implementation of the other Boolean functions of the NP-class, except for the integrase sites which pass from attL/attR to attB/attP. With the characterization of one logic device, we can approximate that we simulate the characterization of all logic devices of the NP-class, only the DNA states will not correspond to the same input states.

It is important to highlight that this simplification is working only if using exclusively inversion-based logic devices. Using excision, the element between integrase sites in reverse orientation will not have the inversion role in gene expression as it will be excised in presence of the input.

Based on this simplification, we reduced the number of devices to implement to one per NP-class, such as 16 for 3 inputs and 380 for 4 inputs. For 3 inputs, the construction and optimization of 16 logic devices for single cell implementation is readily achievable. We proposed the selection of a set of 16 architectures, but a more precise and objective criteria-based selection will be possible based on the Recombinator web interface. Moreover, for logic functions that seem complex to implement, several architectures could be selected in the first round of characterization. From the selected architectures, the selection of integrases, integrase site positions and orientations, and gene expression part identities will have to be performed. To do so, the data from the characterization of integrase sites and terminators of Chapter 2 will be useful. As in this design, inversion is used and thus it required asymmetric terminators, so we are currently characterizing asymmetric terminators in both orientations.

As detailed previously, we can obtain the design of logic devices for the implementation of all Boolean functions in one NP-class from the DNA states of a single logic device. However, the identity of some integrase site pairs will diverge from the DNA states to the logic devices, passing from attL/attR to attB/attP sites. As seen in the characterization of integrase sites in Chapter 2, integrase sites can affect gene expression. Therefore, based on our characterization, we will be able to reduce or at least to predict the change in behavior of the devices from the characterized DNA states due to the change of integrase sites.

Consequently, the characterization and optimization of 16 logic devices will serve as a template for implementing the 218 3-input Boolean functions. The same strategy could be used for 4-input NP-class implementable using only inversion.

Additionally, this characterization and optimization could permit the determination of rules for the design of logic functions responding to an increasing number of inputs.

Chapter 5

Discussion

Contents

5.1 Summary

At the beginning of my thesis, I was interested in increasing the computational power of synthetic biological logic systems. While diving in the literature and in the construction of logic circuits, I realized that the construction of logic circuits having a high computational power will only be useful if they were simple to design and to implement by others scientists. Therefore, I became concerned in providing open-access and straightforward tools for the design and implementation of such logic circuits.

While logic is implementable in living organisms via a large variety of tools, I focused on the implementation of logic using serine integrases. The design of serine-integrase circuits does not fellow any electronic design strategy but permits implementation of compact and single-layer circuits. It was therefore for me an interesting challenge to define systematic and minimized logic strategies for the design of integrase-based logic circuits.

During my thesis, I focused on the implementation of asynchronous Boolean logic circuits and history-dependent logic circuits using two parallel designs. First, I implemented logic in a multicellular system using a systematic design framework and simple and already optimized logic devices. This design strategy makes the design of large logic programs accessible but does not propose the most compact implementation. Therefore, I also worked on the implementation of logic circuits in single-cell systems, pushing the limit of circuit minimization. I believe that this work would permit to increase the computation power of biological logic systems.

5.1.1 Distribution of computation in multicellular system

For implementing multicellular logic systems, I developed a theoretical design framework broadly accessible via a web server: CALIN. For both asynchronous Boolean logic and history-dependent logic, the logic program of interest is decomposed in sub-programs which are implemented in different cellular subpopulation; by composition of these strains the full program is computed.

For Boolean logic, the design is based on a reduced library of logic devices. 14 devices are required for the implementation of all 4-input logic functions. I then worked on the engineering of these 14 devices, so they can be distributed and used by researchers to implement logic circuits in a streamline manner. I engineered these logic devices by decomposition in logic elements. I characterized NOT and IDENTITY elements and composed the well-behaving elements for the design of the devices. Following this process, I obtained in a straightforward manner logic devices with corresponding ON and OFF states and a clear common fold change. While still having background expression level and variable ON states, these devices support composition in a multicellular system. We then characterized the part of the circuits, such as integrase sites and terminators to push forward the optimization by decomposition. We found that integrase sites can have significant promoter and terminator activities. These part characterizations will

be useful for the design of any integrase based circuits and for further optimization of our Boolean logic devices.

For history-dependent logic, the design is based on a scaffold permitting the implementation of all subprograms with gene-expression in a specific lineage of inputs. A specific subprogram is designed by placing genes at GOI positions corresponding to ON input states. We characterized and optimized 2- and 3-input scaffolds by synthesizing recombination intermediate states (OSIRiS method). As a proof-of-concept of the multicellular implementation of history-dependent programs, we implemented a 2-input 2-cell program which behaved as expected. When we have a third inducible integrase, we will characterize a single-cell and a multi-cell 3-input history-dependent program.

The main limitation of this multicellular design is the need to compose a large number of different strains for executing some logic programs. While the different strains can be placed in alginate beads to eliminate potential growth competition and cell-cell communication can be used to integrate the output signal obtaining a constant output level, a high increase number of different strains will still increase the complexity of the implementation of the system. For history-dependent logic, the number of strains is reducible by combining Boolean logic devices with history-dependent devices and minimizing the decomposition of history-dependent programs into programs with a reduced number of inputs. While I implemented a brute-force method to permit combination of Boolean and history-dependent logic devices, systematic history-dependent program minimization methods are required to further push the implementation of history-dependent logic.

5.1.2 Minimization of Boolean logic circuit design.

I also worked on single-cell design with the goal of minimizing the size of logic circuits. Consequently, in this design, a single integrase and a single pair of sites were used per input, pushing forward the circuit compactness. To explore the integrase-circuit design landscape, we generated a complete set of logic devices by an algorithm combining integrase sites, promoters, terminators, and genes. We then obtained numerous designs for each 3-input Boolean function and 91% of 4-input functions. It then proved that all 3-input and most of 4-input Boolean functions are theoretically implementable in single-cell with a single integrase and a single pair of sites per input. The database is available through a web interface called Recombinator. Additionally, a similar method could be used for history-dependent programs.

For Boolean functions, these designs are theoretical and in their generation we did not add any biological constraints, thus we expect that some designs will not behave as predicted. Moreover, as the design is no longer composable, each circuit will need to be optimized in an *ad-hoc* manner. Due to the number of possible designs per functions, I reduced the characterization to a single Boolean function per NP-class, as logic circuits implementing functions from the same

NP-class can be in a first approximation considered equivalent. This reduced the number of 3-input functions to implement and optimize to 16 functions. Consequently, following this strategy, it is possible to prove the possibility of the experimental implementation of all 3-input functions in single-cell.

As we used a brute-force generation method, this strategy is not scalable to a high number of inputs. I aimed during my Ph.D. at formalizing rules for the systematic and scalable design of single-cell logic circuits. As Claude Shannon did for electronic circuits, I wanted to be able to obtain a minimized biological circuit directly from the Boolean function. As seen in Chapter 4.2, the inversion performed by integrase can be associated to the negation of the inputs. Indeed, it seems possible to associate integrase-based logic elements to specific logic operators. I worked on the definition of few systematic design rules and on the development of a Python algorithm that use these rules for the design of logic circuits (Annex A). This work is not complete, and the systematic algorithm that I implemented is still not able to compete with my manual single-cell design. While the previous work permits logic circuits without any biological criteria or limitation, I aimed here at obtaining a design which have more chance to behave as predicted. The implementation of a subset of circuits from the brute-force algorithm will probably permit to new design rules definitions. I hope that the systematic workflow for the design of minimized integrase-based circuits will be achieved one day.

5.1.3 Systematic engineering of synthetic biological circuits.

For integrase-based circuits, the various states of the system are accessible by DNA synthesis as it is encoded in the DNA sequence, unlike repressor-based circuits. Integrase-based circuits can then be optimized by synthesizing DNA intermediate states; this strategy facilitates the debugging of circuit design as seen in Chapter 3.

In general, the construction of synthetic biological circuits by composition of existing biological parts is a fundamental principle of synthetic biology. However, it is still an engineering problem to compose characterized genetic parts for the design of specific genetic circuits. Indeed, the parts behavior are highly dependent on the genetic context and on experimental conditions. For example, in our logic devices, while selecting well-behaving logic elements, we obtained background expression levels and various ON states. Many works have been done to insulate synthetic circuits from biological context [Mutalik 2013a, Mutalik 2013b, Zong 2017, Lou 2012], but it still remains an issue.

Additionally, it seems that randomized DNA sequences can have an effect in gene expression, at least in *E. coli*, such as for the spacer 7 (Chapter 3) and also various integrase sites (Chapter 2). The exact behavior of genetic circuits is still difficult to predict as little is known on the dynamic molecular interaction of DNA and proteins in living organisms.

While the composition of biological parts is still hard to predict, I believe that their charac-

terizations, distributions to the community and the open and transparent publication of results and failures are essential to permit efficient advancement of synthetic biology.

5.2 Control and engineering of serine integrase activity.

The main limitation of this work is the lack of inducible integrases for experimental implementations. Indeed, for multicellular Boolean logic, I characterized all logic devices with constitutive integrase cassettes. Additionally, for history-dependent logic, the characterization of full logic circuits was limited to two inputs.

The engineering of connections between inducible promoters and integrases is not straightforward as leakage in integrase expression leads to an irreversible recombination reaction [Folliard 2015, Courbet 2015a]. Therefore, the integrase expression has to be extremely low in the absence of an inducer while sufficiently high when the inducer is present so that complete recombination is achieved. However, most natural promoters have a high background expression level and/or a reduced fold change of activation. To connect integrases to any inducible promoter, we are developing a systematic engineering workflow that should allow the automation of inducible promoter to integrase connection. To do so, we are screening a set of ribosome binding sites and degradation tags, tuning the translation and degradation levels of the integrase. The variants are tested with the BP target and we select variants with less than 5% switching in the absence of inducer and more than 90% switching in the presence of inducer.

According to our preliminary results, it seems that the switch efficiency is different between inversion and excision of DNA. Excision seems less efficient, at least excision with a reduced spacing (a few dozen base pairs) between pair of integrase sites. This decrease in efficiency is probably due to the physical constraint for the formation of the DNA loop required for the integrase to perform excision. Indeed, for inversion, a simple loop is required to position the sites in the integrase synaptic complex, while two loops are required for excision. Consequently, if the space between the two integrase sites is reduced it might cause a decrease of excision efficiency. In our Boolean logic devices, the space between integrase sites can be as low as 20 base pairs; however, using constitutive integrases we obtained 100% efficiency in the excision due probably of a high quantity of integrases being present. More characterizations have to be performed to confirm these hypotheses. We will characterize inversion and excision with different length spacers between the two integrase sites using a cell-free transcription/translation system, as by adding different quantities of plasmid of integrase we can easily perform a titration of the switch efficiency with the quantity of integrase.

For Boolean logic devices, the reduced recombination efficiency could simplify the engineering of inducible integrases, as more expression leakage would be tolerated by the system. However, this difference of recombination efficiency is problematic for the engineering of integrase cassettes used for inversion and excision, such as for history-dependent logic. The integrase

cassettes have therefore to be characterized in the two conditions. If these hypotheses are confirmed, future logic devices should be designed with a sufficient space between integrase sites in the excision condition.

Nevertheless, we are close to obtain a third integrase switch, after which 3-input history-dependent devices will be characterized. Additionally, we will be able to characterized full 3-input Boolean logic circuits.

We focused for now on induction of integrase via transcription. One limitation of transcription induction is the time required between the detection of the input signal and the folding of the protein, which does not allow for a fast response of the system. To obtain fastest response, integrases could be activated at the post-translational level. One option is to use chemically inducible dimerization. The integrase is split into two domains and each domain is fused to another protein domain that, in presence of the input signal, will induce dimerization of the two fused proteins and therefore activation of the integrase [Weinberg 2017]. Another approach applicable to eukaryotes is to engineer an inducible nuclear localization. In the absence of the input, the integrase is localized in the cytoplasm, and in the presence of input it migrates through the nuclear membrane mediating recombination [Weinberg 2017]. Few systems have been engineered, thus systematic engineering of post-translational activation of integrases needs to be developed to allow for large logic circuits with fast time response.

Despite the use of serine integrases in various genetic circuits, no crystal structure of the integrase in complex with a DNA integrase site exists. Therefore, the recombination mechanism of serine integrases is still not know in detail. If the structure of this complex was known, such as via crystallization or cryoelectron microscopy (CryoEM), it would permit engineering of serine integrases, such as tuning of its catalytic activity, or engineering in a straightforward manner of split-integrase or protein fusion. Additionally, one interesting experiment to study the recombination reaction would be the use of high-speed atomic force microscopy (HS-AFM) to record a "movie" of integrase switching DNA [Ando 2013]. Such work has been performed for Cas9 [Shibata 2017]; the principle would be the same for serine integrases, and it should allow the visualization of the DNA conformations during the excision and inversion recombination reaction. Moreover, it could permit the study of the integrase-excisionase reverse recombination reaction.

5.3 The use of integrase coupled with excisionase permits the implementation of wider types of logic.

During my thesis, I used serine integrases to implement asynchronous, single-shot Boolean and history-dependent logic. Serine integrases mediate irreversible recombination reactions; thus based on integrase only, the logic implemented is asynchronous and single-shot. By combining integrase with a Recombination Directionality Factor (RDF) (excisionase), the reverse recom-

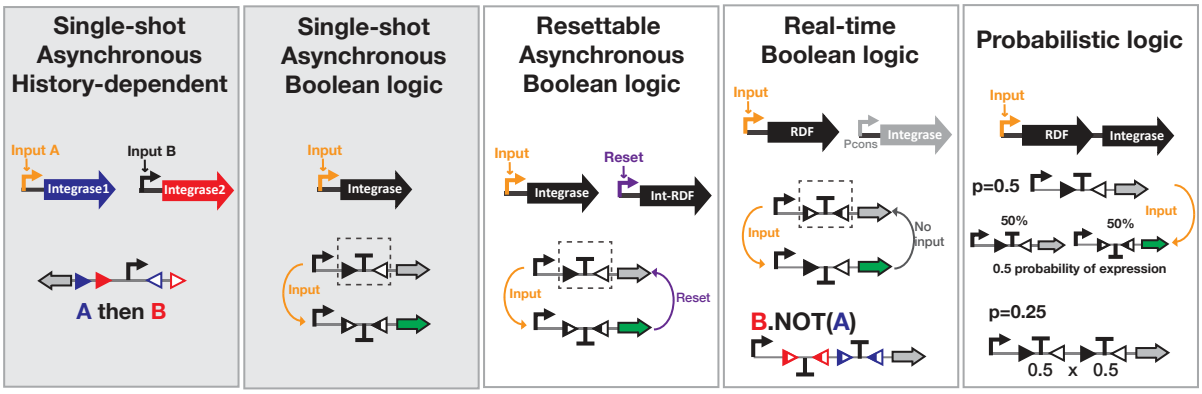


Figure 5.1: **Various integrase-based circuits for the implementation of a large scale of logic.**

bination reaction is performed. Therefore, resettable asynchronous logic is implementable by expressing the integrase with a redirectional factor to perform the reset [Bonnet 2012]. To have a complete reset switch (resettable Boolean logic), the expression of integrases and RDFs have to be stoichiometric [Bonnet 2012]. One simple option to implement stoichiometric expression of integrases and RDFs is to fuse both proteins [Olorunniji 2017]. For now, fusion proteins do not allow a prefect reverse reaction, thus further studies should be performed.

In this design, it is essential to have logic elements based on inversion, as with excision the recombination is destructive and consequently no reset can be performed. For single-cell implementation, logic circuits based exclusively on inversion can be selected through the recombinator web-interface (Chapter 4). For multi-cell implementation, a design is proposed in the supplementary data of the Chapter 2.

In a similar manner, synchronous Boolean logic can be implemented using integrase and RDF. To do so, the RDF is expressed under control of the input signal and the integrase is constitutively expressed. The logic device based on inversion only is composed of attL and attR sites; therefore, the reaction to attB/attP sites is performed in presence of the input with expression of integrase and RDF. In the absence of input, the integrase mediated the reverse reaction, going back to the initial state. The circuit is then synchronous with the presence of inputs. Moreover, probabilistic logic is implementable using integrase and RDF, as has been done based wth DNA computing [Wilhelm 2018]. By tuning the relative expression of integrase and RDF, a switch with a probability of 50% can be engineered corresponding to the implementation of a P-switch. By combining P-switches, various probabilites of expression can be obtained. Guilherme Innocentini and Ovidiu Radulescu developed a stochastic model for the dynamics of this binary biological switch [Innocentini 2016]. This work showed the possibility of using integrase/RDF to implement a p-switch (50% switching probability).

Therefore, the next step for the development of integrase-based logic circuits is the engineering of well-controlled RDF-integrase switches.

5.4 What are the future applications and future challenges of biocomputing?

Serine-integrase based logic circuits have been engineered in *E. coli* [Bonnet 2012, Siuti 2013, Roquet 2016] and in mammalian cells [Weinberg 2017]. Other serine integrase based systems have been implemented in various organisms, such as in *Drosophila* [Bischof 2007], in plants [Hou 2014], and mice [Lakso 1992, Pichel 1993]. Therefore, serine-integrase based logic circuits could be implemented in various organisms, from bacteria to multicellular organisms. The development of such BP targets, integrase cassettes, and logic devices in various model organisms (*Arabidopsis, Drosophila*, Zebrafish, Nematode, Mouse) will extend the use of serine-integrase based circuits and their applications.

Integrases coupled with detection sensors permit signal digitization and amplification, multiplexed signal processing, and data storage in living organisms. It is therefore a great tool for the engineering of biosensor, such as for medical diagnostic [Courbet 2015a]. As it allows signal amplification and digitalization, it is useful for the detection of low concentration of molecules. Additionally, as the output of the system is readable in the DNA, the output can be read by sequencing even if the cell is dead.

I believe that integrase-based logic circuits will serve as a platform for the engineering of upcoming biological-based technology. Likely, the shortest term application of integrase-based circuits will be in medical diagnostics. As research progresses, logic circuits implemented in single-cell systems could be used for engineering therapeutics: e.g. bacteria detecting disease in the human body and producing drugs directly on site.

Moreover, history-dependent logic circuits could be applied over the short term to study time-dependent induction of endogenous signals during development. In general, with the development of accessible tools and design frameworks, all integrase-based circuits could be used to study endogenous signaling networks in living organisms, creating a great tool for fundamental research.

The main interest of integrase-based circuits is their compactness and the irreversibility of switches. Therefore, this technology is complementary to repressor-based circuits that allow implementation of real-time logic. However, efforts have to be made to share and allow more collaborations between labs working on the implementation of logic in living organisms. Moreover, the logic implemented in living organisms does not correspond to what exist in electronics. As the field is growing, it is essential to define common vocabulary for the various types of logic implemented in cells. The use of the terms, sequential logic, asynchronous logic, and history-dependent logic, is not straightforward and often wrongly used due to a lack of proper terms.

To summarize, one challenge for biocomputing and synthetic biology in general is the use

of engineered biological circuits to solve pressing global challenges. Biological logic circuits will serve in the future as a platform to engineer new biologically based technologies.

The implementation of logic within living organism is asking fundamental questions in the logic field. As the implementation of logic circuits in living organisms is scaling up, new logic formalization and logic design minimization have to be developed.

We are now pioneering the development of logic circuits in living organisms building the base of this technology. The coming decades will be exciting with the expansion of biological-based technologies.

Bibliography

[Adleman 1994] L M Adleman. *Molecular computation of solutions to combinatorial problems.* Science, vol. 266, no. 5187, pages 1021–1024, 1994. (Cited on page 27.)

[Akopian 2003] Aram Akopian, Jiuya He, Martin R Boocock and W Marshall Stark. *Chimeric recombinases with designed DNA sequence recognition.* Proc. Natl. Acad. Sci. U. S. A., vol. 100, no. 15, pages 8688–8691, July 2003. (Cited on page 44.)

[Anderson 2006] J Christopher Anderson, Elizabeth J Clarke, Adam P Arkin and Christopher A Voigt. *Environmentally controlled invasion of cancer cells by engineered bacteria.* J. Mol. Biol., vol. 355, no. 4, pages 619–627, January 2006. (Cited on page 17.)

[Anderson 2007] J Christopher Anderson, Christopher A Voigt and Adam P Arkin. *Environmental signal integration by a modular AND gate.* Mol. Syst. Biol., vol. 3, August 2007. (Cited on page 33.)

[Ando 2013] Toshio Ando, Takayuki Uchihashi and Noriyuki Kodera. *High-speed AFM and applications to biomolecular systems.* Annu. Rev. Biophys., vol. 42, pages 393–414, 2013. (Cited on page 168.)

[Annaluru 2014] Narayana Annaluru, Heloise Muller, Leslie a Mitchell, Sivaprakash Ramalingam, Giovanni Stracquadanio, Sarah M Richardson, Jessica S Dymond, Zheng Kuang, Lisa Z Scheifele, Eric M Cooper, Yizhi Cai, Karen Zeller, Neta Agmon, Jeffrey S Han, Michalis Hadjithomas, Jennifer Tullman, Katrina Caravelli, Kimberly Cirelli, Zheyuan Guo, Viktoriya London, Apurva Yeluru, Sindurathy Murugan, Karthikeyan Kandavelou, Nicolas Agier, Gilles Fischer, Kun Yang, J Andrew Martin, Murat Bilgel, Pavlo Bohutskyi, Kristin M Boulier, Brian J Capaldo, Joy Chang, Kristie Charoen, Woo Jin Choi, Peter Deng, James E DiCarlo, Judy Doong, Jessilyn Dunn, Jason I Feinberg, Christopher Fernandez, Charlotte E Floria, David Gladowski, Pasha Hadidi, Isabel Ishizuka, Javaneh Jabbari, Calvin Y L Lau, Pablo a Lee, Sean Li, Denise Lin, Matthias E Linder, Jonathan Ling, Jaime Jonathan Jaime Jonathan Liu, Mariya London, Henry Ma, Jessica Mao, Jessica E McDade, Alexandra McMillan, Aaron M Moore, Won Chan Oh, Yu Ouyang, Ruchi Patel, Marina Paul, Laura C Paulsen, Judy Qiu, Alex Rhee, Matthew G Rubashkin, Ina Y Soh, Nathaniel E Sotuyo, Venkatesh Srinivas, Allison Suarez, Andy Wong, Remus Wong, Wei Rose Xie, Yijie Xu, Allen T Yu, Romain Koszul, Joel S Bader, Jef D Boeke and Srinivasan Chandrasegaran. *Total Synthesis of a Functional Designer Eukaryotic Chromosome.* Science, vol. 344, no. 6179, pages 55–58, 2014. (Cited on page 14.)

[Arber 1962] W Arber and D Dussoix. *Host specificity of DNA produced by Escherichia coli. I. Host controlled modification of bacteriophage lambda.* J. Mol. Biol., vol. 5, pages 18–36, July 1962. (Cited on page 3.)

[Aristotle] Aristotle. Prior analytics. Aeterna Press. (Cited on page 20.)

[Arkin 1994] A Arkin and J Ross. *Computational functions in biochemical reaction networks.* Biophys. J., vol. 67, no. 2, pages 560–578, August 1994. (Cited on page 27.)

[Arkin 1999] Adam P Arkin and Drew Endy. *A Standard Parts List for Biological Circuitry.* DARPA White Paper, pages 1–7, 1999. (Cited on page 6.)

[Arnold 1998] Frances H Arnold. *Design by Directed Evolution.* Acc. Chem. Res., vol. 31, no. 3, pages 125–131, March 1998. (Cited on page 13.)

[Arnold 1999] F H Arnold and A A Volkov. *Directed evolution of biocatalysts.* Curr. Opin. Chem. Biol., vol. 3, no. 1, pages 54–59, February 1999. (Cited on page 13.)

[Aronoff-Spencer 2016] Eliah Aronoff-Spencer, A G Venkatesh, Alex Sun, Howard Brickner, David Looney and Drew A Hall. *Detection of Hepatitis C core antibody by dual-affinity yeast chimera and smartphone-based electrochemical sensing.* Biosens. Bioelectron., vol. 86, pages 690–696, December 2016. (Cited on page 17.)

[Arrault 2000] Alain Arrault. *Les diagrammes de Shao Yong (1012-1077).* Études chinoises, vol. 19, no. 1-2, pages 67–114, 2000. (Cited on page 20.)

[Autebert 1987] Jean-Michel Autebert. Langages algébriques. Masson, 1987. (Cited on page 145.)

[Avery 1944] O T Avery, C M Macleod and M McCarty. *Studies on the chemical nature of the substance inducing formation of pneumococcal types: induction of transformation by a desoxyriboncleic acid fraction isolated from pneumococcus type III.* J. Exp. Med., vol. 79, no. 2, pages 137–158, February 1944. (Cited on page 2.)

[Balagaddé 2008] Frederick K Balagaddé, Hao Song, Jun Ozaki, Cynthia H Collins, Matthew Barnet, Frances H Arnold, Stephen R Quake and Lingchong You. *A synthetic Escherichia coli predator-prey ecosystem.* Mol. Syst. Biol., vol. 4, no. 187, page 187, January 2008. (Cited on page 12.)

[Baldwin 2015] Geoff Baldwin, Travis Bayer, Robert Dickinson, Tom Ellis, Paul S Freemont, Richard I Kitney, Karen Polizzi and Guy-Bart Stan. Synthetic biology — a primer. IMPERIAL COLLEGE PRESS, October 2015. (Cited on page 6.)

[Basu 2004] Subhayu Basu, Rishabh Mehreja, Stephan Thiberge, Ming-Tang Chen and Ron Weiss. *Spatiotemporal control of gene expression with pulse-generating networks.* Proc.

Natl. Acad. Sci. U. S. A., vol. 101, no. 17, pages 6355–6360, April 2004. (Cited on page 208.)

[Basu 2005] Subhayu Basu, Yoram Gerchman, Cynthia H Collins, Frances H Arnold and Ron Weiss. *A synthetic multicellular system for programmed pattern formation.* Nature, vol. 434, no. 7037, pages 1130–1134, 2005. (Cited on page 12.)

[Baumgardner 2009] Jordan Baumgardner, Karen Acker, Oyinade Adefuye, Samuel Thomas Crowley, Will Deloache, James O Dickson, Lane Heard, Andrew T Martens, Nickolaus Morton, Michelle Ritter, Amber Shoecraft, Jessica Treece, Matthew Unzicker, Amanda Valencia, Mike Waters, A Malcolm Campbell, Laurie J Heyer, Jeffrey L Poet and Todd T Eckdahl. *Solving a Hamiltonian Path Problem with a bacterial computer.* J. Biol. Eng., vol. 3, page 11, July 2009. (Cited on page 50.)

[Bayer 2005] Travis S Bayer and Christina D Smolke. *Programmable ligand-controlled riboregulators of eukaryotic gene expression.* Nat. Biotechnol., vol. 23, no. 3, pages 337–343, March 2005. (Cited on page 34.)

[Beaucage 1981] S L Beaucage and M H Caruthers. *Deoxynucleoside phosphoramidites—A new class of key intermediates for deoxypolynucleotide synthesis.* Tetrahedron Lett., vol. 22, no. 20, pages 1859–1862, January 1981. (Cited on page 3.)

[Benenson 2009] Yaakov Benenson. *RNA-based computation in live cells.* Curr. Opin. Biotechnol., vol. 20, no. 4, pages 471–478, August 2009. (Cited on page 34.)

[Bibb 2005] Lori A Bibb, Maria I Hancox and Graham F Hatfull. *Integration and excision by the large serine recombinase phiRv1 integrase.* Mol. Microbiol., vol. 55, no. 6, pages 1896–1910, March 2005. (Cited on page 46.)

[Bischof 2007] Johannes Bischof, Robert K Maeda, Monika Hediger, François Karch and Konrad Basler. *An optimized transgenesis system for Drosophila using germ-line-specific φC31 integrases.* Proc. Natl. Acad. Sci. U. S. A., vol. 104, no. 9, pages 3312–3317, February 2007. (Cited on pages 47 and 170.)

[Boch 2010] Jens Boch and Ulla Bonas. *Xanthomonas AvrBs3 family-type III effectors: discovery and function.* Annu. Rev. Phytopathol., vol. 48, pages 419–436, 2010. (Cited on page 207.)

[Boland 2013] Howard Boland. *Art from synthetic biology.* PhD thesis, 2013. (Cited on page 18.)

[Bonnet 2012] Jerome Bonnet, Pakpoom Subsoontorn and Drew Endy. *Rewritable digital data storage in live cells via engineered control of recombination directionality.* Proc. Natl. Acad. Sci. U. S. A., vol. 109, no. 23, pages 8884–8889, June 2012. (Cited on pages 35, 46, 54, 67, 68, 169, 170 and 209.)

[Bonnet 2013] Jerome Bonnet, Peter Yin, Monica E Ortiz, Pakpoom Subsoontorn and Drew Endy. *Amplifying genetic logic gates.* Science, vol. 340, no. 6132, pages 599–603, 2013. (Cited on pages 12, 32, 35, 50, 67, 68, 71, 111, 128, 138, 139 and 142.)

[Boole 1847] George Boole. The mathematical analysis of logic. Philosophical Library, 1847. (Cited on page 20.)

[Boole 1854] George Boole. An investigation of the laws of thought: On which are founded the mathematical theories of logic and probabilities. Dover Publications, 1854. (Cited on page 20.)

[Boque-Sastre 2015] Raquel Boque-Sastre, Marta Soler, Cristina Oliveira-Mateos, Anna Portela, Catia Moutinho, Sergi Sayols, Alberto Villanueva, Manel Esteller and Sonia Guil. *Head-to-head antisense transcription and R-loop formation promotes transcriptional activation.* Proc. Natl. Acad. Sci. U. S. A., vol. 112, no. 18, pages 5785–5790, May 2015. (Cited on page 149.)

[Bray 1995] D Bray. *Protein molecules as computational elements in living cells*, 1995. (Cited on pages 27 and 111.)

[Breaker 1994] R R Breaker and G F Joyce. *A DNA enzyme that cleaves RNA.* Chem. Biol., vol. 1, no. 4, pages 223–229, December 1994. (Cited on page 28.)

[Breüner 1999] A Breüner, L Brøndsted and K Hammer. *Novel organization of genes involved in prophage excision identified in the temperate lactococcal bacteriophage TP901-1.* J. Bacteriol., vol. 181, no. 23, pages 7291–7297, December 1999. (Cited on page 46.)

[Brøndsted 2001] L Brøndsted, S Ostergaard, M Pedersen, K Hammer and F K Vogensen. *Analysis of the complete DNA sequence of the temperate bacteriophage TP901-1: evolution, structure, and genome organization of lactococcal bacteriophages.* Virology, vol. 283, no. 1, pages 93–109, April 2001. (Cited on page 86.)

[Bryksin 2014] Anton V Bryksin, Ashley C Brown, Michael M Baksh, M G Finn and Thomas H Barker. *Learning from nature - novel synthetic biology approaches for biomaterial design.* Acta Biomater., vol. 10, no. 4, pages 1761–1769, April 2014. (Cited on page 17.)

[Cambray 2013] Guillaume Cambray, Joao C Guimaraes, Vivek K Mutalik, Colin Lam, Quynh-Anh Mai, Tim Thimmaiah, James M Carothers, Adam P Arkin and Drew Endy. *Measurement and modeling of intrinsic transcription terminators.* Nucleic Acids Res., vol. 41, no. 9, page 5139, March 2013. (Cited on page 10.)

[Cameron 2014] D Ewen Cameron, Caleb J Bashor and James J Collins. *A brief history of synthetic biology.* Nat. Rev. Microbiol., vol. 12, no. 5, pages 381–390, 2014. (Cited on pages 2 and 5.)

[Canton 2008] Barry Canton, Anna Labno and Drew Endy. *Refinement and standardization of synthetic biological parts and devices.* Nat. Biotechnol., vol. 26, no. 7, pages 787–793, July 2008. (Cited on page 6.)

[Chalberg 2006] Thomas W Chalberg, Joylette L Portlock, Eric C Olivares, Bhaskar Thyagarajan, Patrick J Kirby, Robert T Hillman, Juergen Hoelters and Michele P Calos. *Integration specificity of phage phiC31 integrase in the human genome.* J. Mol. Biol., vol. 357, no. 1, pages 28–48, March 2006. (Cited on page 48.)

[Chang 2017] Hung-Ju Chang, Peter L Voyvodic, Ana Zuniga and Jerome Bonnet. *Microbially derived biosensors for diagnosis, monitoring and epidemiology.* Microbial biotechnology, August 2017. (Cited on page 17.)

[Chen 2013] Ying-Ja Chen, Peng Liu, Alec A K Nielsen, Jennifer A N Brophy, Kevin Clancy, Todd Peterson and Christopher A Voigt. *Characterization of 582 natural and synthetic terminators and quantification of their design constraints.* Nat. Methods, vol. 10, no. 7, pages 659–664, July 2013. (Cited on pages 10, 70 and 88.)

[Cheo 2004] David L Cheo, Steven A Titus, Devon R N Byrd, James L Hartley, Gary F Temple and Michael A Brasch. *Concerted assembly and cloning of multiple DNA segments using in vitro site-specific recombination: functional analysis of multi-segment expression clones.* Genome Res., vol. 14, no. 10B, pages 2111–2120, October 2004. (Cited on page 47.)

[Chizzolini 2014] Fabio Chizzolini, Michele Forlin, Dario Cecchi and Sheref S Mansy. *Gene position more strongly influences cell-free protein expression from operons than T7 transcriptional promoter strength.* ACS Synth. Biol., vol. 3, no. 6, pages 363–371, June 2014. (Cited on pages 71 and 150.)

[Cobb 2013] Ryan E Cobb, Ning Sun and Huimin Zhao. *Directed evolution as a powerful synthetic biology tool.* Methods, vol. 60, no. 1, pages 81–90, March 2013. (Cited on page 13.)

[Collins 2017] James J Collins and Timothy Kuan-Ta Lu. *Biological analog-to-digital and digital-to-analog converters*, July 2017. (Cited on page 111.)

[Colloms 2014] Sean D Colloms, Christine A Merrick, Femi J Olorunniji, W Marshall Stark, Margaret C M Smith, Anne Osbourn, Jay D Keasling and Susan J Rosser. *Rapid metabolic pathway assembly and modification using serine integrase site-specific recombination.* Nucleic Acids Res., vol. 42, no. 4, page e23, February 2014. (Cited on pages 46, 47, 54, 112 and 142.)

[Courbet 2015a] Alexis Courbet, Drew Endy, Eric Renard, Franck Molina and Jérôme Bonnet. *Detection of pathological biomarkers in human clinical samples via amplifying genetic*

switches and logic gates. Sci. Transl. Med., vol. 7, no. 289, 2015. (Cited on pages 16, 66, 167 and 170.)

[Courbet 2015b] Alexis Courbet, Franck Molina and Patrick Amar. *Computing with synthetic protocells*. Acta Biotheor., vol. 63, no. 3, pages 309–323, September 2015. (Cited on page 30.)

[Courbet 2018] Alexis Courbet, Patrick Amar, François Fages, Eric Renard and Franck Molina. *Computer,Äêaided biochemical programming of synthetic microreactors as diagnostic devices*. Molecular Systems Biology, April 2018. (Cited on page 30.)

[Couturat 1901] L Couturat. La logique de leibniz, d'après des documents inédits. 1901. (Cited on page 20.)

[Couturat 1911] Louis Couturat. The algebra of logic. Open court publishing Company, 1911. (Cited on page 20.)

[Crick 1958] F H Crick. *On protein synthesis*. Symp. Soc. Exp. Biol., vol. 12, pages 138–163, 1958. (Cited on page 2.)

[Davis 2011] Joseph H Davis, Adam J Rubin and Robert T Sauer. *Design, construction and characterization of a set of insulated bacterial promoters*. Nucleic Acids Res., vol. 39, no. 3, pages 1131–1141, February 2011. (Cited on pages 12 and 76.)

[de Lorenzo 2008] Víctor de Lorenzo. *Systems biology approaches to bioremediation*. Curr. Opin. Biotechnol., vol. 19, no. 6, pages 579–589, December 2008. (Cited on page 16.)

[Duan 2010] Faping Duan and John C March. *Engineered bacterial communication prevents Vibrio cholerae virulence in an infant mouse model*. Proc. Natl. Acad. Sci. U. S. A., vol. 107, no. 25, pages 11260–11264, June 2010. (Cited on page 17.)

[Dueber 2003] John E Dueber, Brian J Yeh, Kayam Chak and Wendell A Lim. *Reprogramming control of an allosteric signaling switch through modular recombination*. Science, vol. 301, no. 5641, pages 1904–1908, September 2003. (Cited on page 35.)

[Ede 2016] Christopher Ede, Ximin Chen, Meng-Yin Lin and Yvonne Y Chen. *Quantitative Analyses of Core Promoters Enable Precise Engineering of Regulated Gene Expression in Mammalian Cells*. ACS Synth. Biol., page acssynbio.5b00266, 2016. (Cited on page 9.)

[Egbert 2012] R G Egbert and E Klavins. *Fine-tuning gene networks using simple sequence repeats*. Proceedings of the National Academy of Sciences, vol. 109, no. 42, pages 16817–16822, 2012. (Cited on page 10.)

[Elowitz 2000] Michael B Elowitz and Stanislas Leibler. *A synthetic oscillatory network of transcriptional regulators*. Nature, vol. 403, no. 6767, pages 335–338, 2000. (Cited on pages 12 and 33.)

[Endy 2005] Drew Endy. *Foundations for engineering biology.* Nature, vol. 438, no. 7067, pages 449–453, 2005. (Cited on pages 3 and 5.)

[Endy 2011] Drew Endy. *Drew Endy: Better Computing for the Things We Care About Most.* The New York Times, December 2011. (Cited on page 19.)

[Engler 2008] Carola Engler, Romy Kandzia and Sylvestre Marillonnet. *A One Pot, One Step, Precision Cloning Method with High Throughput Capability.* PLoS One, vol. 3, no. 11, page e3647, 2008. (Cited on page 9.)

[Farinas 2001] E T Farinas, T Bulter and F H Arnold. *Directed enzyme evolution.* Curr. Opin. Biotechnol., vol. 12, no. 6, pages 545–551, December 2001. (Cited on page 13.)

[Feng 1994] J A Feng, R C Johnson and R E Dickerson. *Hin recombinase bound to DNA: the origin of specificity in major and minor groove interactions.* Science, vol. 263, no. 5145, pages 348–355, January 1994. (Cited on page 41.)

[Fiering 1993] S Fiering, C G Kim, E M Epner and M Groudine. *An "in-out" strategy using gene targeting and FLP recombinase for the functional dissection of complex DNA regulatory elements: analysis of the beta-globin locus control region.* Proc. Natl. Acad. Sci. U. S. A., vol. 90, no. 18, pages 8469–8473, September 1993. (Cited on page 47.)

[Folliard 2015] Thomas Folliard, Jerome Bonnet, John Ward, Frank Baganz, Christopher Grant and Drew Endy. Connecting boolean integrase logic gates to a novel alkane control signal via engineered level matching. November 2015. (Cited on page 167.)

[Friedland 2009] A E Friedland, T K Lu, X Wang, D Shi, G Church and J J Collins. *Synthetic Gene Networks That Count.* Science, vol. 324, no. 5931, pages 1199–1202, 2009. (Cited on pages 54 and 111.)

[Friedman 1986] Arthur D Friedman. Fundamentals of logic design and switching theory. Computer Science Press, 1986. (Cited on page 156.)

[Gaber 2014] Rok Gaber, Tina Lebar, Andreja Majerle, Branko Šter, Andrej Dobnikar, Mojca Benčina and Roman Jerala. *Designable DNA-binding domains enable construction of logic circuits in mammalian cells.* Nat. Chem. Biol., vol. 10, no. 3, pages 203–208, March 2014. (Cited on pages 33 and 207.)

[Galanie 2015] S Galanie, K Thodey, I J Trenchard, M Filsinger Interrante and C D Smolke. *Complete biosynthesis of opioids in yeast.* Science, vol. 349, no. 6252, pages 1095–1100, 2015. (Cited on page 15.)

[Gander 2017] Miles W Gander, Justin D Vrana, William E Voje, James M Carothers and Eric Klavins. *Digital logic circuits in yeast with CRISPR-dCas9 NOR gates.* Nat. Commun., vol. 8, page 15459, May 2017. (Cited on pages 31, 33 and 208.)

[Gardner 2000] T S Gardner, C R Cantor and J J Collins. *Construction of a genetic toggle switch in Escherichia coli.* Nature, vol. 403, no. 6767, pages 339–342, January 2000. (Cited on pages 10, 12, 32 and 111.)

[Ghosh 2006] Pallavi Ghosh, Laura R Wasil and Graham F Hatfull. *Control of phage Bxb1 excision by a novel recombination directionality factor.* PLoS Biol., vol. 4, no. 6, page e186, June 2006. (Cited on page 46.)

[Ghosh 2008] Pallavi Ghosh, Lori A Bibb and Graham F Hatfull. *Two-step site selection for serine-integrase-mediated excision: DNA-directed integrase conformation and central dinucleotide proofreading.* Proc. Natl. Acad. Sci. U. S. A., vol. 105, no. 9, pages 3238–3243, March 2008. (Cited on page 46.)

[Gibson 2008] Daniel G Gibson, Gwynedd A Benders, Cynthia Andrews-Pfannkoch, Evgeniya A Denisova, Holly Baden-Tillson, Jayshree Zaveri, Timothy B Stockwell, Anushka Brownley, David W Thomas, Mikkel A Algire, Chuck Merryman, Lei Young, Vladimir N Noskov, John I Glass, J Craig Venter, Clyde A Hutchison 3rd and Hamilton O Smith. *Complete chemical synthesis, assembly, and cloning of a Mycoplasma genitalium genome.* Science, vol. 319, no. 5867, pages 1215–1220, February 2008. (Cited on page 13.)

[Gibson 2009] Daniel G Gibson, Lei Young, Ray-Yuan Chuang, J Craig Venter, Clyde A Hutchison and Hamilton O Smith. *Enzymatic assembly of DNA molecules up to several hundred kilobases.* Nat. Methods, vol. 6, no. 5, pages 343–345, 2009. (Cited on page 9.)

[Gibson 2010] D G Gibson, J I Glass, C Lartigue, V N Noskov, R-Y Chuang, M A Algire, G A Benders, M G Montague, L Ma, M M Moodie, C Merryman, S Vashee, R Krishnakumar, N Assad-Garcia, C Andrews-Pfannkoch, E A Denisova, L Young, Z-Q Qi, T H Segall-Shapiro, C H Calvey, P P Parmar, C A Hutchison, H O Smith and J C Venter. *Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome.* Science, vol. 329, no. 5987, pages 52–56, 2010. (Cited on page 13.)

[Glass 2006] John I Glass, Nacyra Assad-Garcia, Nina Alperovich, Shibu Yooseph, Matthew R Lewis, Mahir Maruf, Clyde A Hutchison 3rd, Hamilton O Smith and J Craig Venter. *Essential genes of a minimal bacterium.* Proc. Natl. Acad. Sci. U. S. A., vol. 103, no. 2, pages 425–430, January 2006. (Cited on page 14.)

[Goeddel 1979] D V Goeddel, D G Kleid, F Bolivar, H L Heyneker, D G Yansura, R Crea, T Hirose, A Kraszewski, K Itakura and A D Riggs. *Expression in Escherichia coli of chemically synthesized genes for human insulin.* Proc. Natl. Acad. Sci. U. S. A., vol. 76, no. 1, pages 106–110, January 1979. (Cited on page 3.)

[Gopfrich 2018] Kerstin Gopfrich, Ilia Platzman and Joachim P Spatz. *Mastering Complexity: Towards Bottom-up Construction of Multifunctional Eukaryotic Synthetic Cells.* Trends Biotechnol., vol. 0, no. 0, April 2018. (Cited on page 14.)

[Green 2017] Alexander A Green, Jongmin Kim, Duo Ma, Pamela A Silver, James J Collins and Peng Yin. *Complex cellular logic computation using ribocomputing devices*. Nature, vol. 548, no. 7665, pages 117–121, August 2017. (Cited on pages 32 and 34.)

[Grindley 2006] Nigel D F Grindley, Katrine L Whiteson and Phoebe A Rice. *Mechanisms of site-specific recombination*. Annu. Rev. Biochem., vol. 75, pages 567–605, 2006. (Cited on page 41.)

[Guerrier-Takada 1983] C Guerrier-Takada, K Gardiner, T Marsh, N Pace and S Altman. *The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme*. Cell, vol. 35, no. 3 Pt 2, pages 849–857, December 1983. (Cited on page 28.)

[Guet 2002] Călin C Guet, Michael B Elowitz, Weihong Hsing and Stanislas Leibler. *Combinatorial synthesis of genetic networks*. Science, vol. 296, no. 5572, pages 1466–1470, May 2002. (Cited on page 33.)

[Guiziou 2016] Sarah Guiziou, Vincent Sauveplane, Hung-Ju Chang, Caroline Clerté, Nathalie Declerck, Matthieu Jules and Jerome Bonnet. *A part toolbox to tune genetic expression in Bacillus subtilis*. Nucleic Acids Res., vol. 44, no. 15, pages 7495–7508, September 2016. (Cited on pages 10 and 11.)

[Guo 1997] F Guo, D N Gopaul and G D van Duyne. *Structure of Cre recombinase complexed with DNA in a site-specific recombination synapse*. Nature, vol. 389, no. 6646, pages 40–46, September 1997. (Cited on page 43.)

[Ham 2006] Timothy S Ham, Sung Kuk Lee, Jay D Keasling and Adam P Arkin. *A Tightly Regulated Inducible Expression System Utilizing the fim Inversion Recombination Switch*. Biotechnology, vol. 10, pages 1–4, 2006. (Cited on page 111.)

[Ham 2008a] Timothy S Ham, Sung K Lee, Jay D Keasling and Adam P Arkin. *Design and Construction of a Double Inversion Recombination Switch for Heritable Sequential Genetic Memory*. PLoS One, vol. 3, no. 7, page e2815, July 2008. (Cited on page 52.)

[Ham 2008b] Timothy S Ham, Sung K Lee, Jay D Keasling and Adam P Arkin. *Design and Construction of a Double Inversion Recombination Switch for Heritable Sequential Genetic Memory*. PLoS One, vol. 3, no. 7, page e2815, July 2008. (Cited on page 112.)

[Hartley 2000] J L Hartley, G F Temple and M A Brasch. *DNA cloning using in vitro site-specific recombination*. Genome Res., vol. 10, no. 11, pages 1788–1795, November 2000. (Cited on page 47.)

[Hartwell 1999] L H Hartwell, J J Hopfield, S Leibler and A W Murray. *From molecular to modular cell biology*. Nature, vol. 402, no. 6761 Suppl, pages C47–52, December 1999. (Cited on page 3.)

[Haynes 2008] Karmella a Haynes, Marian L Broderick, Adam D Brown, Trevor L Butner, James O Dickson, W Lance Harden, Lane H Heard, Eric L Jessen, Kelly J Malloy, Brad J Ogden, Sabriya Rosemond, Samantha Simpson, Erin Zwack, a Malcolm Campbell, Todd T Eckdahl, Laurie J Heyer and Jeffrey L Poet. *Engineering bacteria to solve the Burnt Pancake Problem.* J. Biol. Eng., vol. 2, page 8, 2008. (Cited on pages 50 and 51.)

[Heichman 1990] K A Heichman and R C Johnson. *The Hin invertasome: protein-mediated joining of distant recombination sites at the enhancer.* Science, vol. 249, no. 4968, pages 511–517, August 1990. (Cited on page 45.)

[Hjelmfelt 1991] A Hjelmfelt, E D Weinberger and J Ross. *Chemical implementation of neural networks and Turing machines.* Proc. Natl. Acad. Sci. U. S. A., vol. 88, no. 24, pages 10983–10987, December 1991. (Cited on pages 27 and 29.)

[Hjelmfelt 1992] A Hjelmfelt, E D Weinberger and J Ross. *Chemical implementation of finite-state machines.* Proc. Natl. Acad. Sci. U. S. A., vol. 89, no. 1, pages 383–387, January 1992. (Cited on page 29.)

[Hjelmfelt 1993] A Hjelmfelt, F W Schneider and J Ross. *Pattern recognition in coupled chemical kinetic systems.* Science, vol. 260, no. 5106, pages 335–337, April 1993. (Cited on page 27.)

[Hou 2014] Lili Hou, Yuan-Yeu Yau, Junjie Wei, Zhiguo Han, Zhicheng Dong and David W Ow. *An open-source system for in planta gene stacking by Bxb1 and Cre recombinases.* Mol. Plant, vol. 7, no. 12, pages 1756–1765, December 2014. (Cited on page 170.)

[Hsiao 2016] Victoria Hsiao, Yutaka Hori, Paul Wk Rothemund and Richard M Murray. *A population-based temporal logic gate for timing and recording chemical events.* Mol. Syst. Biol., vol. 12, no. 5, page 869, May 2016. (Cited on pages 36, 53, 112 and 209.)

[Hutchison 2016] C A Hutchison, R-Y Chuang, V N Noskov, N Assad-Garcia, T J Deerinck, M H Ellisman, J Gill, K Kannan, B J Karas, L Ma, J F Pelletier, Z-Q Qi, R A Richter, E A Strychalski, L Sun, Y Suzuki, B Tsvetanova, K S Wise, H O Smith, J I Glass, C Merryman, D G Gibson and J C Venter. *Design and synthesis of a minimal bacterial genome.* Science, vol. 351, no. 6280, pages aad6253–aad6253, 2016. (Cited on page 14.)

[Innocentini 2016] Guilherme C P Innocentini, Sarah Guiziou, Jerome Bonnet and Ovidiu Radulescu. *Analytic framework for a stochastic binary biological switch.* Phys Rev E, vol. 94, no. 6-1, page 062413, December 2016. (Cited on page 169.)

[Isaacs 2004] Farren J Isaacs, Daniel J Dwyer, Chunming Ding, Dmitri D Pervouchine, Charles R Cantor and James J Collins. *Engineered riboregulators enable post-transcriptional control of gene expression.* Nat. Biotechnol., vol. 22, no. 7, pages 841–847, July 2004. (Cited on pages 32 and 34.)

[Itakura 1977] K Itakura, T Hirose, R Crea, A D Riggs, H L Heyneker, F Bolivar and H W Boyer. *Expression in Escherichia coli of Chemically Synthesized.* Science, vol. 198, no. 4321, pages 1056–1063, December 1977. (Cited on page 3.)

[Jaakko T. Astola 2006] Radomir S Stankovi Jaakko T. Astola. Fundamentals of switching theory and logic design. 2006. (Cited on page 156.)

[Jackson 1972] D A Jackson, R H Symons and Paul Berg. *Biochemical method for inserting new genetic information into DNA of Simian Virus 40: circular SV40 DNA molecules containing lambda phage genes and the ....* Proceedings of the, 1972. (Cited on page 3.)

[Jacob 1961] Francois Jacob and Jacques Monod. *Genetic Regulatory Mechanisms in the Synthesis of Proteins.* pages 433–471, 1961. (Cited on pages 19 and 32.)

[Jiang 2018] Jianjuan Jiang, Shaoqing Zhang, Zhigang Qian, Nan Qin, Wenwen Song, Long Sun, Zhitao Zhou, Zhifeng Shi, Liang Chen, Xinxin Li, Ying Mao, David L Kaplan, Stephanie N Gilbert Corder, Xinzhong Chen, Mengkun Liu, Fiorenzo G Omenetto, Xiaoxia Xia and Tiger H Tao. *Protein Bricks: 2D and 3D Bio-Nanostructures with Shape and Function on Demand.* Adv. Mater., page e1705919, March 2018. (Cited on pages 17 and 18.)

[Joung 1994] J K Joung, D M Koepp and A Hochschild. *Synergistic activation of transcription by bacteriophage lambda cI protein and E. coli cAMP receptor protein.* Science, vol. 265, no. 5180, pages 1863–1866, September 1994. (Cited on page 32.)

[Jusiak 2016] Barbara Jusiak, Sara Cleto, Pablo Perez-Piñera and Timothy K Lu. *Engineering Synthetic Gene Circuits in Living Cells with CRISPR Technology.* Trends Biotechnol., vol. 34, no. 7, pages 535–547, July 2016. (Cited on page 208.)

[Kalos 2013] Michael Kalos and Carl H June. *Adoptive T cell transfer for cancer immunotherapy in the era of synthetic biology.* Immunity, vol. 39, no. 1, pages 49–60, July 2013. (Cited on page 17.)

[Kelly 2009] Jason R Kelly, Adam J Rubin, Joseph H Davis, Caroline M Ajo-Franklin, John Cumbers, Michael J Czar, Kim de Mora, Aaron L Glieberman, Dileep D Monie and Drew Endy. *Measuring the activity of BioBrick promoters using an in vivo reference standard.* J. Biol. Eng., vol. 3, no. 1, page 4, 2009. (Cited on page 6.)

[Keravala 2006] Annahita Keravala, Amy C Groth, Sohail Jarrahian, Bhaskar Thyagarajan, Jason J Hoyt, Patrick J Kirby and Michele P Calos. *A diversity of serine phage integrases mediate site-specific recombination in mammalian cells.* Mol. Genet. Genomics, vol. 276, no. 2, pages 135–146, August 2006. (Cited on page 46.)

[Khaleel 2011] Thanafez Khaleel, Ellen Younger, Andrew R McEwan, Anpu S Varghese and Margaret C M Smith. *A phage protein that binds φC31 integrase to switch its directionality.* Mol. Microbiol., vol. 80, no. 6, pages 1450–1463, June 2011. (Cited on page 46.)

[Kiani 2014] Samira Kiani, Jacob Beal, Mohammad R Ebrahimkhani, Jin Huh, Richard N Hall, Zhen Xie, Yinqing Li and Ron Weiss. *CRISPR transcriptional repression devices and layered circuits in mammalian cells.* Nat. Methods, vol. 11, no. 7, pages 723–726, July 2014. (Cited on pages 33 and 208.)

[Kim 2006] Jongmin Kim, Kristin S White and Erik Winfree. *Construction of an in vitro bistable circuit from synthetic transcriptional switches.* Mol. Syst. Biol., vol. 2, page 68, December 2006. (Cited on page 29.)

[Kim 2011] Jongmin Kim and Erik Winfree. *Synthetic in vitro transcriptional oscillators.* Mol. Syst. Biol., vol. 7, page 465, February 2011. (Cited on page 29.)

[Knight 2003] Tom Knight. *Idempotent vector design for standard assembly of biobricks.* Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 2003. (Cited on pages 6 and 8.)

[Kosuri 2013] Sriram Kosuri, Daniel B Goodman, Guillaume Cambray, Vivek K Mutalik, Yuan Gao, Adam P Arkin, Drew Endy and George M Church. *Composability of regulatory sequences controlling transcription and translation in Escherichia coli.* Proc. Natl. Acad. Sci. U. S. A., vol. 110, no. 34, pages 14024–14029, 2013. (Cited on page 10.)

[Kramer 2004] Beat P Kramer, Cornelius Fischer and Martin Fussenegger. *BioLogic gates enable logical transcription control in mammalian cells.* Biotechnol. Bioeng., vol. 87, no. 4, pages 478–484, August 2004. (Cited on page 33.)

[Krink-Koutsoubelis 2018] Nicolas Krink-Koutsoubelis, Anne C Loechner, Anna Lechner, Hannes Link, Charles M Denby, Bastian Vögeli, Tobias J Erb, Satoshi Yuzawa, Tadas Jakociunas, Leonard Katz, Michael K Jensen, Victor Sourjik and Jay D Keasling. *Engineered Production of Short-Chain Acyl-Coenzyme A Esters in Saccharomyces cerevisiae.* ACS Synth. Biol., vol. 7, no. 4, pages 1105–1115, April 2018. (Cited on page 15.)

[Kruger 1982] K Kruger, P J Grabowski, A J Zaug, J Sands, D E Gottschling and T R Cech. *Self-splicing RNA: autoexcision and autocyclization of the ribosomal RNA intervening sequence of Tetrahymena.* Cell, vol. 31, no. 1, pages 147–157, November 1982. (Cited on page 28.)

[Kumar 1992] V Kumar. *Algorithms for constraint-satisfaction problems: A survey.* AI magazine, 1992. (Cited on page 152.)

[Kumari 2008] Anjali Kumari, Patrizia Pasini and Sylvia Daunert. *Detection of bacterial quorum sensing N-acyl homoserine lactones in clinical samples.* Anal. Bioanal. Chem., vol. 391, no. 5, pages 1619–1627, July 2008. (Cited on page 17.)

[Lajoie 2013a] M J Lajoie, S Kosuri, J A Mosberg, C J Gregg, D Zhang and G M Church. *Probing the limits of genetic recoding in essential genes.* Science, vol. 342, no. 6156, pages 361–363, October 2013. (Cited on page 13.)

[Lajoie 2013b] Marc J Lajoie, Alexis J Rovner, Daniel B Goodman, Hans-Rudolf Aerni, Adrian D Haimovich, Gleb Kuznetsov, Jaron A Mercer, Harris H Wang, Peter A Carr, Joshua A Mosberg, Nadin Rohland, Peter G Schultz, Joseph M Jacobson, Jesse Rinehart, George M Church and Farren J Isaacs. *Genomically recoded organisms expand biological functions.* Science, vol. 342, no. 6156, pages 357–360, October 2013. (Cited on page 13.)

[Lakso 1992] M Lakso, B Sauer, B Mosinger Jr, E J Lee, R W Manning, S H Yu, K L Mulder and H Westphal. *Targeted oncogene activation by site-specific recombination in transgenic mice.* Proc. Natl. Acad. Sci. U. S. A., vol. 89, no. 14, pages 6232–6236, July 1992. (Cited on pages 48 and 170.)

[Lederman 2006] Harvey Lederman, Joanne Macdonald, Darko Stefanovic and Milan N Stojanovic. *Deoxyribozyme-based three-input logic gates and construction of a molecular full adder.* Biochemistry, vol. 45, no. 4, pages 1194–1199, January 2006. (Cited on page 28.)

[Leduc 1910] Stéphane Leduc. Théorie physico-chimique de la vie et générations spontanées, volume 1. Poinat, 1910. (Cited on page 2.)

[Leduc 1912] Stéphane Leduc. La biologie synthétique, volume 2. A. Poinat, 1912. (Cited on page 2.)

[Lee 1998] G Lee and I Saito. *Role of nucleotide sequences of loxP spacer region in Cre-mediated recombination.* Gene, vol. 216, no. 1, pages 55–65, August 1998. (Cited on page 43.)

[Lee 2008] Sung Kuk Lee, Howard Chou, Timothy S Ham, Taek Soon Lee and Jay D Keasling. *Metabolic engineering of microorganisms for biofuels production: from bugs to synthetic biology to fuels.* Curr. Opin. Biotechnol., vol. 19, no. 6, pages 556–563, December 2008. (Cited on page 15.)

[Lee 2015] Michael E Lee, William C DeLoache, Bernardo Cervantes and John E Dueber. *A Highly Characterized Yeast Toolkit for Modular, Multipart Assembly.* ACS Synth. Biol., vol. 4, no. 9, pages 975–986, 2015. (Cited on page 9.)

[Lehner 2017] Benjamin A E Lehner, Dominik T Schmieden and Anne S Meyer. *A Straightforward Approach for 3D Bacterial Printing.* ACS Synth. Biol., vol. 6, no. 7, pages 1124–1130, July 2017. (Cited on page 18.)

[Li 2005] Weikai Li, Satwik Kamtekar, Yong Xiong, Gary J Sarkis, Nigel D F Grindley and Thomas A Steitz. *Structure of a synaptic gammadelta resolvase tetramer covalently linked to two cleaved DNAs.* Science, vol. 309, no. 5738, pages 1210–1215, August 2005. (Cited on page 44.)

[Lienert 2013] Florian Lienert, Joseph P Torella, Jan-Hung Chen, Michael Norsworthy, Ryan R Richardson and Pamela A Silver. *Two- and three-input TALE-based AND logic computation in embryonic stem cells.* Nucleic Acids Res., vol. 41, no. 21, pages 9967–9975, November 2013. (Cited on pages 33 and 207.)

[Lipton 1995] R J Lipton. *DNA solution of hard computational problems.* Science, vol. 268, no. 5210, pages 542–545, April 1995. (Cited on page 28.)

[Livet 2007] Jean Livet, Tamily A Weissman, Hyuno Kang, Ryan W Draft, Ju Lu, Robyn A Bennis, Joshua R Sanes and Jeff W Lichtman. *Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system.* Nature, vol. 450, no. 7166, pages 56–62, November 2007. (Cited on pages 48, 50 and 51.)

[Lohmueller 2012] Jason J Lohmueller, Thomas Z Armel and Pamela A Silver. *A tunable zinc finger-based framework for Boolean logic computation in mammalian cells.* Nucleic Acids Res., vol. 40, no. 11, pages 5180–5187, June 2012. (Cited on pages 33 and 207.)

[Lou 2010] Chunbo Lou, Xili Liu, Ming Ni, Yiqi Huang, Qiushi Huang, Longwen Huang, Lingli Jiang, Dan Lu, Mingcong Wang, Chang Liu, Daizhuo Chen, Chongyi Chen, Xiaoyue Chen, Le Yang, Haisu Ma, Jianguo Chen and Qi Ouyang. *Synthesizing a novel genetic sequential logic circuit: a push-on push-off switch.* Mol. Syst. Biol., vol. 6, pages 1–11, March 2010. (Cited on page 111.)

[Lou 2012] Chunbo Lou, Brynne Stanton, Ying-Ja Chen, Brian Munsky and Christopher A Voigt. *Ribozyme-based insulator parts buffer synthetic circuits from genetic context.* Nat. Biotechnol., vol. 30, no. 11, pages 1137–1142, 2012. (Cited on pages 12, 70, 121 and 166.)

[Lucks 2011] Julius B Lucks, Lei Qi, Vivek K Mutalik, Denise Wang and Adam P Arkin. *Versatile RNA-sensing transcriptional regulators for engineering genetic networks.* Proc. Natl. Acad. Sci. U. S. A., vol. 108, no. 21, pages 8617–8622, May 2011. (Cited on page 34.)

[Macia 2014] Javier Macia and Ricard Sole. *How to make a synthetic multicellular computer.* PLoS One, vol. 9, no. 2, page e81248, February 2014. (Cited on pages 33 and 37.)

[Macia 2016] Javier Macia, Romilde Manzoni, Núria Conde, Arturo Urrios, Eulàlia de Nadal, Ricard Solé and Francesc Posas. *Implementation of Complex Biological Logic Circuits Using Spatially Distributed Multicellular Consortia*. PLoS Comput. Biol., vol. 12, no. 2, page e1004685, February 2016. (Cited on pages 33, 36, 37 and 38.)

[Maeder 2008] Morgan L Maeder, Stacey Thibodeau-Beganny, Anna Osiak, David A Wright, Reshma M Anthony, Magdalena Eichtinger, Tao Jiang, Jonathan E Foley, Ronnie J Winfrey, Jeffrey A Townsend, Erica Unger-Wallace, Jeffry D Sander, Felix Müller-Lerch, Fengli Fu, Joseph Pearlberg, Carl Göbel, Justin P Dassie, Shondra M Pruett-Miller, Matthew H Porteus, Dennis C Sgroi, A John Iafrate, Drena Dobbs, Paul B McCray, Toni Cathomen, Daniel F Voytas and J Keith Joung. *Rapid "Open-Source" Engineering of Customized Zinc-Finger Nucleases for Highly Efficient Gene Modification*. Mol. Cell, vol. 31, no. 2, pages 294–301, July 2008. (Cited on page 207.)

[Manber 1988] Udi Manber. *Using Induction to Design Algorithms*. Commun. ACM, vol. 31, no. 11, pages 1300–1313, November 1988. (Cited on page 151.)

[Mandali 2017] Sridhar Mandali, Kushol Gupta, Anthony R Dawson, Gregory D Van Duyne and Reid C Johnson. *Control of Recombination Directionality by the Listeria Phage A118 Protein Gp44 and the Coiled-Coil Motif of Its Serine Integrase*. J. Bacteriol., vol. 199, no. 11, June 2017. (Cited on page 46.)

[Mano 2014] Morris Mano and Charles Kime. Logic and computer design fundamentals. 2014. (Cited on page 26.)

[McBride 1983] L J McBride and M H Caruthers. *An investigation of several deoxynucleoside phosphoramidites useful for synthesizing deoxyoligonucleotides*. Tetrahedron Lett., vol. 24, no. 3, pages 245–248, January 1983. (Cited on page 3.)

[McCulloch 1943] Warren S McCulloch and Walter Pitts. *A logical calculus of the ideas immanent in nervous activity*. Bull. Math. Biophys., vol. 5, no. 4, pages 115–133, December 1943. (Cited on page 26.)

[Mercy 2017] Guillaume Mercy, Julien Mozziconacci, Vittore F Scolari, Kun Yang, Guanghou Zhao, Agnès Thierry, Yisha Luo, Leslie A Mitchell, Michael Shen, Yue Shen, Roy Walker, Weimin Zhang, Yi Wu, Ze-Xiong Xie, Zhouqing Luo, Yizhi Cai, Junbiao Dai, Huanming Yang, Ying-Jin Yuan, Jef D Boeke, Joel S Bader, Héloïse Muller and Romain Koszul. *3D organization of synthetic and scrambled chromosomes*. Science, vol. 355, no. 6329, March 2017. (Cited on page 14.)

[Mitchell 1998] Melanie Mitchell. *Computation in Cellular Automata: A Selected Review*. Plan. Perspect., vol. 95, page 140, 1998. (Cited on page 27.)

[Miyamoto 2012] Takafumi Miyamoto, Robert DeRose, Allison Suarez, Tasuku Ueno, Melinda Chen, Tai-Ping Sun, Michael J Wolfgang, Chandrani Mukherjee, David J Meyers and Takanari Inoue. *Rapid and orthogonal logic gating with a gibberellin-induced dimerization system.* Nat. Chem. Biol., vol. 8, no. 5, pages 465–470, March 2012. (Cited on page 35.)

[Monod 1961] Jacques Monod and François Jacob. *General conclusions: teleonomic mechanisms in cellular metabolism, growth, and differentiation.* Cold Spring Harbor Laboratory Press, vol. 26, no. Cold Spring Harbor Symposia on Quantitative Biology, pages 389–401, 1961. (Cited on page 2.)

[Montagne 2011] Kevin Montagne, Raphael Plasson, Yasuyuki Sakai, Teruo Fujii and Yannick Rondelez. *Programming an in vitro DNA oscillator using a molecular networking strategy.* Mol. Syst. Biol., vol. 7, page 466, February 2011. (Cited on page 29.)

[Moon 2012] Tae Seok Moon, Chunbo Lou, Alvin Tamsir, Brynne C Stanton and Christopher A Voigt. *Genetic programs constructed from layered logic gates in single cells.* Nat. Commun., vol. 491, no. 7423, pages 249–253, 2012. (Cited on pages 31 and 33.)

[Mouw 2008] Kent W Mouw, Sally-J Rowland, Mark M Gajjar, Martin R Boocock, W Marshall Stark and Phoebe A Rice. *Architecture of a serine recombinase-DNA regulatory complex.* Mol. Cell, vol. 30, no. 2, pages 145–155, April 2008. (Cited on page 44.)

[Mullis 1986] K Mullis, F Faloona, S Scharf, R Saiki, G Horn and H Erlich. *Specific enzymatic amplification of DNA in vitro: the polymerase chain reaction.* Cold Spring Harb. Symp. Quant. Biol., vol. 51 Pt 1, pages 263–273, 1986. (Cited on page 3.)

[Mutalik 2013a] V K Mutalik, J C Guimaraes, G Cambray, Q A Mai, M J Christoffersen, L Martin, A Yu, C Lam, C Rodriguez, G Bennett, J D Keasling, D Endy and A P Arkin. *Quantitative estimation of activity and quality for collections of functional genetic elements.* Nat.Methods, vol. 10, no. 1548-7105 (Electronic), pages 347–353, 2013. (Cited on pages 9, 10, 12 and 166.)

[Mutalik 2013b] Vivek K Mutalik, Joao C Guimaraes, Guillaume Cambray, Colin Lam, Marc Juul Christoffersen, Quynh-Anh Mai, Andrew B Tran, Morgan Paull, Jay D Keasling, Adam P Arkin and Drew Endy. *Precise and reliable gene expression via standard transcription and translation initiation elements.* Nat. Methods, vol. 10, no. 4, pages 354–360, 2013. (Cited on pages 12, 70, 76 and 166.)

[Nash 1981] H A Nash. *Integration and Excision of Bacteriophage λ: The Mechanism of Conservative Site Specific Recombination.* Annu. Rev. Genet., vol. 15, no. 1, pages 143–167, December 1981. (Cited on pages 40 and 43.)

[Nielsen 2014] Alec A K Nielsen and Christopher A Voigt. *Multi-input CRISPR/Cas genetic circuits that interface host regulatory networks.* Mol. Syst. Biol., vol. 10, page 763, November 2014. (Cited on pages 33 and 208.)

[Nielsen 2016] Alec A K Nielsen, Bryan S Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A Strychalski, David Ross, Douglas Densmore and Christopher A Voigt. *Genetic circuit design automation.* Science, vol. 352, no. 6281, page aac7341, April 2016. (Cited on pages 10, 12, 31, 33, 36, 37, 38, 66, 71 and 112.)

[Olivares 2002] Eric C Olivares, Roger P Hollis, Thomas W Chalberg, Leonard Meuse, Mark A Kay and Michele P Calos. *Site-specific genomic integration produces therapeutic Factor IX levels in mice.* Nat. Biotechnol., vol. 20, no. 11, pages 1124–1128, November 2002. (Cited on page 48.)

[Olorunniji 2017] Femi J Olorunniji, Arlene L McPherson, Susan J Rosser, Margaret C M Smith, Sean D Colloms and W Marshall Stark. *Control of serine integrase recombination directionality by fusion with the directionality factor.* Nucleic Acids Res., June 2017. (Cited on pages 46 and 169.)

[Ortiz-Urda 2002] Susana Ortiz-Urda, Bhaskar Thyagarajan, Douglas R Keene, Qun Lin, Min Fang, Michele P Calos and Paul A Khavari. *Stable nonviral genetic correction of inherited human skin disease.* Nat. Med., vol. 8, no. 10, pages 1166–1170, October 2002. (Cited on page 48.)

[Ostrov 2016] Nili Ostrov, Matthieu Landon, Marc Guell, Gleb Kuznetsov, Jun Teramoto, Natalie Cervantes, Minerva Zhou, Kerry Singh, Michael G Napolitano, Mark Moosburner, Ellen Shrock, Benjamin W Pruitt, Nicholas Conway, Daniel B Goodman, Cameron L Gardner, Gary Tyree, Alexandra Gonzales, Barry L Wanner, Julie E Norville, Marc J Lajoie and George M Church. *Design, synthesis, and testing toward a 57-codon genome.* Science, vol. 353, no. 6301, pages 819–822, August 2016. (Cited on page 13.)

[Paddon 2013] C J Paddon, P J Westfall, D J Pitera, K Benjamin, K Fisher, D McPhee, M D Leavell, A Tai, A Main, D Eng, D R Polichuk, K H Teoh, D W Reed, T Treynor, J Lenihan, M Fleck, S Bajad, G Dang, D Dengrove, D Diola, G Dorin, K W Ellens, S Fickes, J Galazzo, S P Gaucher, T Geistlinger, R Henry, M Hepp, T Horning, T Iqbal, H Jiang, L Kizer, B Lieu, D Melis, N Moss, R Regentin, S Secrest, H Tsuruta, R Vazquez, L F Westblade, L Xu, M Yu, Y Zhang, L Zhao, J Lievense, P S Covello, J D Keasling, K K Reiling, N S Renninger and J D Newman. *High-level semi-synthetic production of the potent antimalarial artemisinin.* Nature, vol. 496, no. 7446, pages 528–532, April 2013. (Cited on page 15.)

[Padirac 2012] Adrien Padirac, Teruo Fujii and Yannick Rondelez. *Bottom-up construction of in vitro switchable memories.* Proc. Natl. Acad. Sci. U. S. A., vol. 109, no. 47, pages E3212–20, November 2012. (Cited on page 29.)

[Padirac 2013] Adrien Padirac, Teruo Fujii and Yannick Rondelez. *Nucleic acids for the rational design of reaction circuits.* Curr. Opin. Biotechnol., vol. 24, no. 4, pages 575–580, August 2013. (Cited on page 28.)

[Pardee 2014] Keith Pardee, Alexander A Green, Tom Ferrante, D Ewen Cameron, Ajay DaleyKeyser, Peng Yin and James J Collins. *Paper-Based Synthetic Gene Networks.* Cell, vol. 159, no. 4, pages 940–954, 2014. (Cited on page 17.)

[Pardee 2016] Keith Pardee, Alexander A Green, Melissa K Takahashi, Dana Braff, Guillaume Lambert, Jeong Wook Lee, Tom Ferrante, Duo Ma, Nina Donghia, Melina Fan, Nichole M Daringer, Irene Bosch, Dawn M Dudley, David H O'Connor, Lee Gehrke and James J Collins. *Rapid, Low-Cost Detection of Zika Virus Using Programmable Biomolecular Components.* Cell, vol. 165, no. 5, pages 1255–1266, May 2016. (Cited on pages 17 and 29.)

[Pichel 1993] J G Pichel, M Lakso and H Westphal. *Timing of SV40 oncogene activation by site-specific recombination determines subsequent tumor progression during murine lens development.* Oncogene, vol. 8, no. 12, pages 3333–3342, December 1993. (Cited on pages 48 and 170.)

[Podhajska 1985] A J Podhajska, N Hasan and W Szybalski. *Control of cloned gene expression by promoter inversion in vivo: construction of the heat-pulse-activated att-nutL-p-att-N module.* Gene, vol. 40, no. 1, pages 163–168, 1985. (Cited on pages 48, 50 and 111.)

[Privman 2010] Vladimir Privman, Jian Zhou, Jan Halámek and Evgeny Katz. *Realization and properties of biochemical-computing biocatalytic XOR gate based on signal change.* J. Phys. Chem. B, vol. 114, no. 42, pages 13601–13608, October 2010. (Cited on page 29.)

[Qi 2014] Lei S Qi and Adam P Arkin. *A versatile framework for microbial engineering using synthetic non-coding RNAs.* Nat. Rev. Microbiol., vol. 12, no. 5, pages 341–354, 2014. (Cited on page 34.)

[Qian 2011a] L Qian and E Winfree. *Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades.* Science, vol. 332, no. 6034, pages 1196–1201, 2011. (Cited on page 29.)

[Qian 2011b] Lulu Qian, Erik Winfree and Jehoshua Bruck. *Neural network computation with DNA strand displacement cascades.* Nature, vol. 475, no. 7356, pages 368–372, July 2011. (Cited on page 29.)

[Reed 1981] R R Reed and N D Grindley. *Transposon-mediated site-specific recombination in vitro: DNA cleavage and protein-DNA linkage at the recombination site.* Cell, vol. 25, no. 3, pages 721–728, September 1981. (Cited on page 44.)

[Reed 1984] R R Reed and C D Moser. *Resolvase-mediated recombination intermediates contain a serine residue covalently linked to DNA.* Cold Spring Harb. Symp. Quant. Biol., vol. 49, pages 245–249, 1984. (Cited on page 44.)

[Regot 2010] Sergi Regot, Javier Macía, Núria Conde, Kentaro Furukawa, Jimmy Kjellén, Tom Peeters, Stefan Hohmann, Eulàlia de Nadal, Francesc Posas and Ricard Solé. *Distributed biological computation with multicellular engineered networks*. Nature, vol. 469, no. 7329, pages 207–211, December 2010. (Cited on pages 33 and 37.)

[Renella 2016] Giancarlo Renella and Laura Giagnoni. *Light dazzles from the black box: whole-cell biosensors are ready to inform on fundamental soil biological processes*. Chemical and Biological Technologies in Agriculture, vol. 3, no. 1, page 8, March 2016. (Cited on page 16.)

[Rhodius 2013] Virgil A Rhodius, Thomas H Segall-Shapiro, Brian D Sharon, Amar Ghodasara, Ekaterina Orlova, Hannah Tabakh, David H Burkhardt, Kevin Clancy, Todd C Peterson, Carol A Gross and Christopher A Voigt. *Design of orthogonal genetic switches based on a crosstalk map of σs, anti-σs, and promoters*. Mol. Syst. Biol., vol. 9, page 702, October 2013. (Cited on page 33.)

[Richardson 2017] Sarah M Richardson, Leslie A Mitchell, Giovanni Stracquadanio, Kun Yang, Jessica S Dymond, James E DiCarlo, Dongwon Lee, Cheng Lai Victor Huang, Srinivasan Chandrasegaran, Yizhi Cai, Jef D Boeke and Joel S Bader. *Design of a synthetic yeast genome*. Science, vol. 355, no. 6329, pages 1040–1044, March 2017. (Cited on page 14.)

[Rinaudo 2007] Keller Rinaudo, Leonidas Bleris, Rohan Maddamsetti, Sairam Subramanian, Ron Weiss and Yaakov Benenson. *A universal RNAi-based logic evaluator that operates in mammalian cells*. Nat. Biotechnol., vol. 25, no. 7, pages 795–801, July 2007. (Cited on page 34.)

[Ro 2006] Dae-Kyun Ro, Eric M Paradise, Mario Ouellet, Karl J Fisher, Karyn L Newman, John M Ndungu, Kimberly A Ho, Rachel A Eachus, Timothy S Ham, James Kirby, Michelle C Y Chang, Sydnor T Withers, Yoichiro Shiba, Richmond Sarpong and Jay D Keasling. *Production of the antimalarial drug precursor artemisinic acid in engineered yeast*. vol. 440, no. April, pages 3–6, 2006. (Cited on page 15.)

[Roquet 2016] Nathaniel Roquet, Ava P Soleimany, Alyssa C Ferris, Scott Aaronson and Timothy K Lu. *Synthetic recombinase-based state machines in living cells*. Science, vol. 353, no. 6297, page aad8559, July 2016. (Cited on pages 36, 55, 112, 170 and 209.)

[Ruder 2011] Warren C Ruder, Ting Lu and James J Collins. *Synthetic biology moving into the clinic*. Science, vol. 333, no. 6047, pages 1248–1252, 2011. (Cited on page 16.)

[Rutherford 2013] Karen Rutherford, Peng Yuan, Kay Perry, Robert Sharp and Gregory D Van Duyne. *Attachment site recognition and regulation of directionality by the serine integrases*. Nucleic Acids Res., vol. 41, no. 17, pages 8341–8356, September 2013. (Cited on page 46.)

[Sauer 1988] B Sauer and N Henderson. *The cyclization of linear DNA in Escherichia coli by site-specific recombination.* Gene, vol. 70, no. 2, pages 331–341, October 1988. (Cited on page 47.)

[Sauer 1994] B Sauer. *Site-specific recombination: developments and applications.* Curr. Opin. Biotechnol., vol. 5, no. 5, pages 521–527, October 1994. (Cited on page 47.)

[Schlake 1994] T Schlake and J Bode. *Use of mutated FLP recognition target (FRT) sites for the exchange of expression cassettes at defined chromosomal loci.* Biochemistry, vol. 33, no. 43, pages 12746–12751, November 1994. (Cited on page 48.)

[Schmieden 2016] Dominik T Schmieden, Anne S Meyer and Marie-Eve Aubin-Tam. *Using bacteria to make improved, nacre-inspired materials.* MRS Advances, vol. 1, no. 8, pages 559–564, 2016. (Cited on page 18.)

[Schwander 2016] Thomas Schwander, Lennart Schada von Borzyskowski, Simon Burgener, Niña Socorro Cortina and Tobias J Erb. *A synthetic pathway for the fixation of carbon dioxide in vitro.* Science, vol. 354, no. 6314, pages 900–904, November 2016. (Cited on page 15.)

[Scientific 2017] Invitrogen by Thermofisher Scientific. *User guide - Jump-In CHO-K1 Retargeting Kit*, 2017. (Cited on page 48.)

[Seelig 2006] Georg Seelig, David Soloveichik, David Yu Zhang and Erik Winfree. *Enzyme-free nucleic acid logic circuits.* Science, vol. 314, no. 5805, pages 1585–1588, December 2006. (Cited on page 29.)

[Shannon 1936] Claude Elwood Shannon. *A symbolic analysis of relay and switching circuits.* PhD thesis, Massachusetts Institute of Technology, 1936. (Cited on page 21.)

[Shen 2016] Yue Shen, Giovanni Stracquadanio, Yun Wang, Kun Yang, Leslie A Mitchell, Yaxin Xue, Yizhi Cai, Tai Chen, Jessica S Dymond, Kang Kang, Jianhui Gong, Xiaofan Zeng, Yongfen Zhang, Yingrui Li, Qiang Feng, Xun Xu, Jun Wang, Jian Wang, Huanming Yang, Jef D Boeke and Joel S Bader. *SCRaMbLE generates designed combinatorial stochastic diversity in synthetic chromosomes.* Genome Res., vol. 26, no. 1, pages 36–49, January 2016. (Cited on pages 14 and 48.)

[Shendure 2017] Jay Shendure, Shankar Balasubramanian, George M Church, Walter Gilbert, Jane Rogers, Jeffery A Schloss and Robert H Waterston. *DNA sequencing at 40: past, present and future.* Nature, vol. 550, no. 7676, pages 345–353, October 2017. (Cited on page 8.)

[Shibata 2017] Mikihiro Shibata, Hiroshi Nishimasu, Noriyuki Kodera, Seiichi Hirano, Toshio Ando, Takayuki Uchihashi and Osamu Nureki. *Real-space and real-time dynamics of*

CRISPR-Cas9 visualized by high-speed atomic force microscopy. Nat. Commun., vol. 8, no. 1, page 1430, November 2017. (Cited on page 168.)

[Shipman 2016] S L Shipman, Seth L Shipman, Jeff Nivala, Jeffrey D Macklis and George M Church. *Molecular recordings by directed CRISPR spacer acquisition.* Science, vol. 1175, pages 1–16, 2016. (Cited on page 209.)

[Shis 2013] David L Shis and Matthew R Bennett. *Library of synthetic transcriptional AND gates built with split T7 RNA polymerase mutants.* Proc. Natl. Acad. Sci. U. S. A., vol. 110, no. 13, pages 5028–5033, March 2013. (Cited on pages 31 and 33.)

[Siuti 2013] Piro Siuti, John Yazbek and Timothy K Lu. *Synthetic circuits integrating logic and memory in living cells.* Nat. Biotechnol., vol. 31, no. 5, pages 448–452, May 2013. (Cited on pages 12, 35, 50, 111, 142 and 170.)

[Sleight 2013] Sean C Sleight and Herbert M Sauro. *Visualization of evolutionary stability dynamics and competitive fitness of Escherichia coli engineered with randomized multigene circuits.* ACS Synth. Biol., vol. 2, no. 9, pages 519–528, September 2013. (Cited on page 112.)

[Smith 1973] H O Smith and D Nathans. *Letter: A suggested nomenclature for bacterial host modification and restriction systems and their enzymes.* J. Mol. Biol., vol. 81, no. 3, pages 419–423, December 1973. (Cited on page 3.)

[Smith 2003] Hamilton O Smith, Clyde A Hutchison, Cynthia Pfannkoch and J Craig Venter. *Generating a synthetic genome by whole genome assembly: phiX174 bacteriophage from synthetic oligonucleotides.* Proc. Natl. Acad. Sci. U. S. A., vol. 100, no. 26, pages 15440–15445, December 2003. (Cited on page 13.)

[Smolke 2009] Christina D Smolke. *Building outside of the box: iGEM and the BioBricks Foundation.* Nat. Biotechnol., vol. 27, no. 12, pages 1099–1102, December 2009. (Cited on page 6.)

[Stanton 2014] Brynne C Stanton, Alec A K Nielsen, Alvin Tamsir, Kevin Clancy, Todd Peterson and Christopher A Voigt. *Genomic mining of prokaryotic repressors for orthogonal logic gates.* Nat. Chem. Biol., vol. 10, no. 2, pages 99–105, February 2014. (Cited on pages 31 and 33.)

[Stark 1989] W M Stark, D J Sherratt and M R Boocock. *Site-specific recombination by Tn3 resolvase: topological changes in the forward and reverse reactions.* Cell, vol. 58, no. 4, pages 779–790, August 1989. (Cited on page 40.)

[Stojanovic 2002] Milan N Stojanovic, Tiffany Elizabeth Mitchell and Darko Stefanovic. *Deoxyribozyme-based logic gates.* J. Am. Chem. Soc., vol. 124, no. 14, pages 3555–3561, April 2002. (Cited on page 28.)

[Stojanovic 2003] Milan N Stojanovic and Darko Stefanovic. *A deoxyribozyme-based molecular automaton.* Nat. Biotechnol., vol. 21, no. 9, pages 1069–1074, September 2003. (Cited on page 28.)

[Struhl 1993] G Struhl and K Basler. *Organizing activity of wingless protein in Drosophila.* Cell, vol. 72, no. 4, pages 527–540, February 1993. (Cited on page 48.)

[Subsoontorn 2012a] Pakpoom Subsoontorn and Drew Endy. *Design and Analysis of Genetically Encoded Counters.* Procedia Comput. Sci., vol. 11, pages 43–54, January 2012. (Cited on pages 54 and 209.)

[Subsoontorn 2012b] Pakpoom Subsoontorn, Jongmin Kim and Erik Winfree. *Ensemble Bayesian analysis of bistability in a synthetic transcriptional switch.* ACS Synth. Biol., vol. 1, no. 8, pages 299–316, August 2012. (Cited on page 29.)

[Subsoontorn 2014] Pakpoom Subsoontorn. *Reliable Functional Composition of a Recombinase Device Family.* no. August, 2014. (Cited on pages 35 and 209.)

[Szybalski 1978] W Szybalski and A Skalka. *Nobel prizes and restriction enzymes.* Gene, vol. 4, no. 3, pages 181–182, November 1978. (Cited on page 3.)

[Tabor 2009] Jeffrey J Tabor, Howard M Salis, Zachary Booth Simpson, Aaron A Chevalier, Anselm Levskaya, Edward M Marcotte, Christopher A Voigt and Andrew D Ellington. *A synthetic genetic edge detection program.* Cell, vol. 137, no. 7, pages 1272–1281, June 2009. (Cited on page 12.)

[Tamsir 2011] Alvin Tamsir, Jeffrey J Tabor and Christopher A Voigt. *Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'.* Nature, vol. 469, no. 7329, pages 212–215, 2011. (Cited on page 33.)

[Tang 2018] Weixin Tang and David R Liu. *Rewritable multi-event analog recording in bacterial and mammalian cells.* Science, February 2018. (Cited on page 209.)

[Toman 1985] Z Toman, C Dambly-Chaudière, L Tenenbaum and M Radman. *A system for detection of genetic and epigenetic alterations in Escherichia coli induced by DNA-damaging agents.* J. Mol. Biol., vol. 186, no. 1, pages 97–105, November 1985. (Cited on pages 12 and 111.)

[Townshend 2015] Brent Townshend, Andrew B Kennedy, Joy S Xiang and Christina D Smolke. *High-throughput cellular RNA device engineering.* Nat. Methods, vol. 12, no. 10, pages 989–994, October 2015. (Cited on page 34.)

[Uesaka 2014] Masahiro Uesaka, Osamu Nishimura, Yasuhiro Go, Kinichi Nakashima, Kiyokazu Agata and Takuya Imamura. *Bidirectional promoters are the major source of gene activation-associated non-coding RNAs in mammals.* BMC Genomics, vol. 15, page 35, January 2014. (Cited on page 149.)

[Umeno 2004] Daisuke Umeno and Frances H Arnold. *Evolution of a pathway to novel long-chain carotenoids.* J. Bacteriol., vol. 186, no. 5, pages 1531–1536, March 2004. (Cited on page 13.)

[Urrios 2016] Arturo Urrios, Javier Macia, Romilde Manzoni, Núria Conde, Adriano Bonforti, Eulàlia de Nadal, Francesc Posas and Ricard Solé. *A Synthetic Multicellular Memory Device.* ACS Synth. Biol., vol. 5, no. 8, pages 862–873, August 2016. (Cited on pages 33, 37 and 209.)

[Urrios 2018] Arturo Urrios, Eva Gonzalez-Flo, David Canadell, Eulàlia de Nadal, Javier Macia and Francesc Posas. *Plug-and-Play Multicellular Circuits with Time-Dependent Dynamic Responses.* ACS Synth. Biol., April 2018. (Cited on pages 66 and 209.)

[Van Duyne 2013] Gregory D Van Duyne and Karen Rutherford. *Large serine recombinase domain structure and attachment site binding.* Crit. Rev. Biochem. Mol. Biol., vol. 48, no. 5, pages 476–491, September 2013. (Cited on page 46.)

[Vidali 2001] Vidali. *Bioremediation. An overview.* Pure Appl. Chem., vol. 73, no. 7, pages 1163–1172, 2001. (Cited on page 16.)

[Von Neumann 1996] John Von Neumann and Arthur Walter Burks. Theory of self-reproducing automata. University of Illinois Press Urbana, 1996. (Cited on page 27.)

[Wang 2006] Yongzhong Wang, Hyeon-Joo Kim, Gordana Vunjak-Novakovic and David L Kaplan. *Stem cell-based tissue engineering with silk biomaterials.* Biomaterials, vol. 27, no. 36, pages 6064–6082, December 2006. (Cited on page 17.)

[Wang 2009] Harris H Wang, Farren J Isaacs, Peter A Carr, Zachary Z Sun, George Xu, Craig R Forest and George M Church. *Programming cells by multiplex genome engineering and accelerated evolution.* Nature, vol. 460, no. 7257, pages 894–898, August 2009. (Cited on page 13.)

[Wang 2011] Baojun Wang, Richard I Kitney, Nicolas Joly and Martin Buck. *Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology.* Nat. Commun., vol. 2, page 508, October 2011. (Cited on pages 31 and 33.)

[Watson 1953] James D Watson, Francis H C Crick and Others. *Molecular structure of nucleic acids.* Nature, vol. 171, no. 4356, pages 737–738, 1953. (Cited on page 2.)

[Weinberg 2017] Benjamin H Weinberg, N T Hang Pham, Leidy D Caraballo, Thomas Lozanoski, Adrien Engel, Swapnil Bhatia and Wilson W Wong. *Large-scale design of robust genetic circuits with multiple inputs and outputs for mammalian cells.* Nat. Biotechnol., vol. 35, no. 5, pages 453–462, May 2017. (Cited on pages 36, 52, 55, 66, 111, 142, 168 and 170.)

[Weiss 2001] Ron Weiss, T Knight and Gerald Sussman. *Cellular computation and communication using engineered genetic regulatory networks.* Cellular computing, pages 120–121, 2001. (Cited on page 12.)

[Wilhelm 2018] Daniel Wilhelm, Jehoshua Bruck and Lulu Qian. *Probabilistic switching circuits in DNA.* Proc. Natl. Acad. Sci. U. S. A., vol. 115, no. 5, pages 903–908, January 2018. (Cited on page 169.)

[Win 2007] Maung Nyan Win and Christina D Smolke. *A modular and extensible RNA-based gene-regulatory platform for engineering cellular function.* Proc. Natl. Acad. Sci. U. S. A., vol. 104, no. 36, pages 14283–14288, September 2007. (Cited on page 34.)

[Win 2008] M N Win and C D Smolke. *Higher-Order Cellular Information Processing with Synthetic RNA Devices.* Science, vol. 322, no. 5900, pages 456–460, October 2008. (Cited on page 34.)

[Wolf 2008] Denise M Wolf, Lisa Fontaine-Bodin, Ilka Bischofs, Gavin Price, Jay Keasling and Adam P Arkin. *Memory in Microbes: Quantifying History-Dependent Behavior in a Bacterium.* PLoS One, vol. 3, no. 2, page e1700, February 2008. (Cited on page 111.)

[Wolpert 2015] Lewis Wolpert, Cheryll Tickle and Alfonso Martinez Arias. Principles of development. Oxford University Press, 2015. (Cited on page 111.)

[Xiang 2006] Shuanglin Xiang, Johannes Fruehauf and Chiang J Li. *Short hairpin RNA–expressing bacteria elicit RNA interference in mammals.* Nat. Biotechnol., vol. 24, page 697, May 2006. (Cited on page 17.)

[Xie 2011] Zhen Xie, Liliana Wroblewska, Laura Prochazka, Ron Weiss and Yaakov Benenson. *Multi-Input RNAi-Based Logic Circuit For Indentification of Specifc Cancer Cells.* Science, vol. 333, no. 2011, pages 1307–1312, 2011. (Cited on page 34.)

[Yang 1995] W Yang and T A Steitz. *Crystal structure of the site-specific recombinase gamma delta resolvase complexed with a 34 bp cleavage site.* Cell, vol. 82, no. 2, pages 193–207, July 1995. (Cited on page 44.)

[Yang 2014] Lei Yang, Alec A K Nielsen, Jesus Fernandez-Rodriguez, Conor J McClune, Michael T Laub, Timothy K Lu and Christopher A Voigt. *Permanent genetic memory with >1-byte capacity.* Nat. Methods, vol. 11, no. 12, pages 1261–1266, 2014. (Cited on pages 46, 52, 67, 68 and 76.)

[Yuan 2008] Peng Yuan, Kushol Gupta and Gregory D Van Duyne. *Tetrameric structure of a serine integrase catalytic domain.* Structure, vol. 16, no. 8, pages 1275–1286, August 2008. (Cited on page 46.)

[Yurke 2000] B Yurke, A J Turberfield, A P Mills Jr, F C Simmel and J L Neumann. *A DNA-fuelled molecular machine made of DNA.* Nature, vol. 406, no. 6796, pages 605–608, August 2000. (Cited on page 28.)

[Zadorin 2017] Anton S Zadorin, Yannick Rondelez, Guillaume Gines, Vadim Dilhas, Georg Urtel, Adrian Zambrano, Jean-Christophe Galas and André Estevez-Torres. *Synthesis and materialization of a reaction-diffusion French flag pattern.* Nat. Chem., vol. 9, no. 10, pages 990–996, October 2017. (Cited on page 29.)

[Zalatan 2015] Jesse G Zalatan, Michael E Lee, Ricardo Almeida, Luke A Gilbert, Evan H Whitehead, Marie La Russa, Jordan C Tsai, Jonathan S Weissman, John E Dueber, Lei S Qi and Wendell A Lim. *Engineering complex synthetic transcriptional programs with CRISPR RNA scaffolds.* Cell, vol. 160, no. 1-2, pages 339–350, January 2015. (Cited on page 208.)

[Zhang 2011] Lin Zhang, Guoping Zhao and Xiaoming Ding. *Tandem assembly of the epothilone biosynthetic gene cluster by in vitro site-specific recombination.* Sci. Rep., vol. 1, page 141, November 2011. (Cited on page 47.)

[Zhang 2013] Lin Zhang, Binyan Zhu, Ruixue Dai, Guoping Zhao and Xiaoming Ding. *Control of directionality in Streptomyces phage φBT1 integrase-mediated site-specific recombination.* PLoS One, vol. 8, no. 11, page e80434, November 2013. (Cited on page 46.)

[Zhang 2014] Haoqian Zhang, Min Lin, Handuo Shi, Weiyue Ji, Longwen Huang, Xiaomeng Zhang, Shan Shen, Rencheng Gao, Shuke Wu, Chengzhe Tian, Zhenglin Yang, Guosheng Zhang, Siheng He, Hao Wang, Tiffany Saw, Yiwei Chen and Qi Ouyang. *Programming a Pavlovian-like conditioning circuit in Escherichia coli.* Nat. Commun., vol. 5, page 3102, 2014. (Cited on page 111.)

[Zhou 2009] Jian Zhou, Mary A Arugula, Jan Halámek, Marcos Pita and Evgeny Katz. *Enzyme-based NAND and NOR logic gates with modular design.* J. Phys. Chem. B, vol. 113, no. 49, pages 16065–16070, December 2009. (Cited on page 29.)

[Zhu 2014] Fangfang Zhu, Matthew Gamboa, Alfonso P Farruggio, Simon Hippenmeyer, Bosiljka Tasic, Birgitt Schüle, Yanru Chen-Tsai and Michele P Calos. *DICE, an efficient system for iterative genomic editing in human pluripotent stem cells.* Nucleic Acids Res., vol. 42, no. 5, page e34, March 2014. (Cited on page 48.)

[Zong 2017] Yeqing Zong, Haoqian M Zhang, Cheng Lyu, Xiangyu Ji, Junran Hou, Xian Guo, Qi Ouyang and Chunbo Lou. *Insulated transcriptional elements enable precise design of genetic circuits.* Nat. Commun., vol. 8, no. 1, page 52, July 2017. (Cited on pages 12 and 166.)

Appendix

Annex A

Systematic rules for designing minimized integrase logic circuits.

In Chapter 4, I used a brute force approach to obtain all possible gate designs and then determined the logic function implemented by each design. We have found that all functions belonging to similar P- and NP-classes could be attained by sampling the recombinatorial space of a given device in single cells. However, we are still left with many design possibilities for implementing a particular NP-class, none of them being very strictly constrained by biologically informed design rules. While the brute force approach works well for 2- and 3-input devices, it starts to reach its limits for 4 inputs and beyond, for two reasons. First, after generating all possible devices for 4-input devices, we were able to realize 91% of the total number of functions in single-cell. Second, and more importantly, the brute force approach requires a lot of computational resources, and lot of storage capacity due to the number of generated sequences, and these computational needs increase exponentially with the number of inputs. Consequently, this approach will not be possible for an increasing number of inputs.

In Chapter 2, I automatized the design of logic functions in multicellular systems by decomposition into subprograms. To do so, I established systematic rules for function decomposition and direct design of logic devices from sub-functions.

The main limitation of the design presented in Chapter 2 is the number of required strains. Here, I aim at defining systematic design rules that permit the implementation of logic in a minimized number of cells.

As Claude Shannon did for electronic circuits, I want to be able to obtain minimized biological circuits directly from Boolean functions. As seen in Chapter 4.2, the inversion performed by an integrase can be associated to the negation of the inputs. Indeed, it seems possible to associate integrase-based logic elements to specific logic operators. I worked on the definition of few systematic design rules and on the development of a Python algorithm that uses these rules for the design of logic circuits.

The Python algorithm developed here allows for the reduction of the number of strains required for the systematic implementation of Boolean functions, going to a maximum of two strains for three inputs.

This work is not finished, and the systematic algorithm that I implemented is still not able to compete with my manual single-cell design skills. However, I hope that this work could be pushed forward, and by analyzing designs generated in Chapter 4.1, new design rules could be determined.

A.1 Definition of elements and composition rules for the design of single-cell logic devices

In Chapter 2, logic devices are based on the composition of elements exclusively using excision. Here, I extended the set of logic elements to the use of excision and inversion, therefore to four additional elements (Figure A.1). NOT and IDENTITY functions are therefore implementable in three different manners. Here, I consider that the output gene is placed in the 3' of the construct and a promoter is added in the 5' of the construct if no promoter is present in the construction.

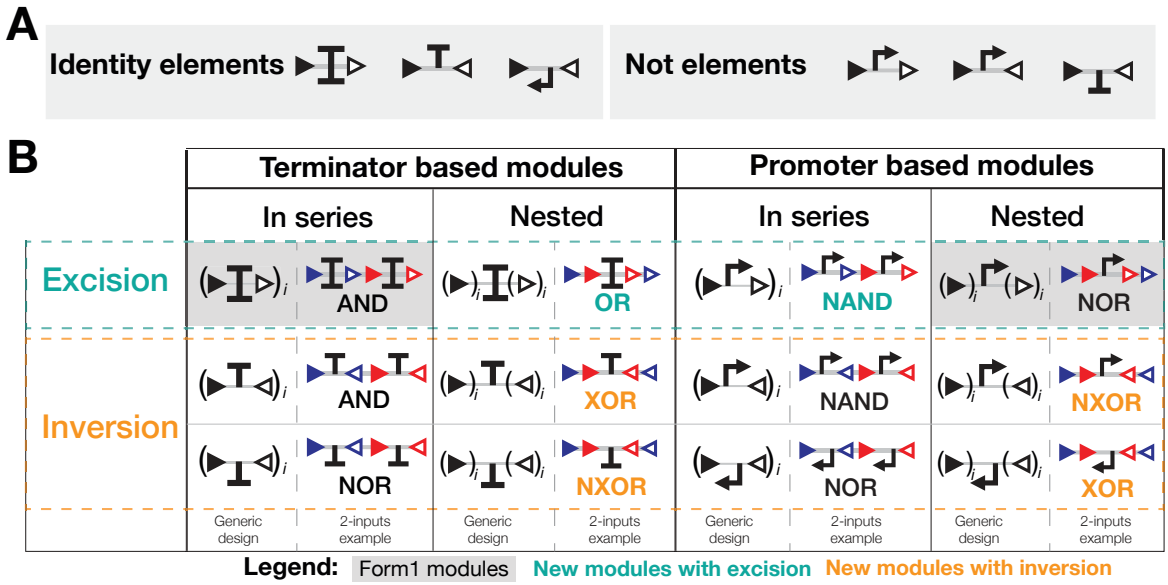


Figure A.1: **Definition of elements and modules corresponding to the implementation of a large set of logic functions.** (A) Definition of a set of IDENTITY and NOT elements based on promoters, terminators, excisions, and inversions. (B) Composition of elements in series or nested. For each module, the generic design is represented on the left and a 2-input example with the logic function implemented on the right. The Form 1 modules correspond to the modules used in Chapter 2 for the multicellular implementation.

Two identical elements responding to different inputs are composable either in series or in parallel (corresponding to the nested sites). These two biological compositions correspond to different logic operations depending on if the element is based on promoters or terminators.

(1) The composition of terminator-based modules in series corresponds to the conjunction of the elementary functions (X $\wedge$ Y).

(2) The composition of promoter-based modules in series corresponds to the disjunction of the elementary functions (X ∨ Y).

(3) The composition of terminator-based modules in parallel (nested) corresponds to the disjunction of the elementary functions if based on excision (X ∨ Y) and to the exclusive disjunction of the elementary functions if based on inversion (X ⊻ Y).

(4) The composition of promoter-based modules in parallel (nested) corresponds to the conjunction of the elementary functions if based on excision (X ∧ Y) and to the exclusive conjunction of the elementary functions if based on inversion (X $\overline{\wedge}$ Y).

The composition of identical elements responding to different inputs corresponds to a module. I obtained twelve different modules from these six elements. These modules permit the implementation of AND, NOR, OR, NAND, XOR, and NXOR 2-input gates and these designs are scalable to N-inputs following the generic design detailed in Figure A.1B.

Then, I defined a reduced set of module compositions (Figure A.2A). Terminator-based modules are composable in series and correspond to the conjunction of the functions. Promoter-based modules are composable in series and correspond to the disjunction of the functions. A terminator-based module is composable in the 3' end of a promoter-based module and corresponds to the conjunction of the two functions.

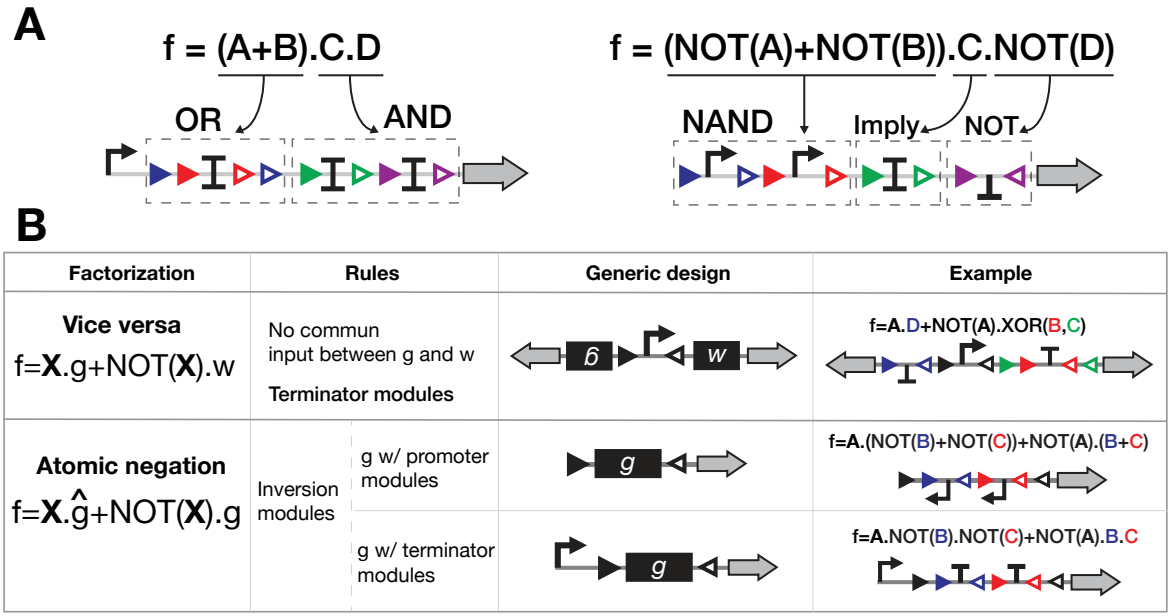


Figure A.2: **Rules of composition of modules and factorization rules.** (A) Examples of composition of modules. (B) Vice versa and atomic negation factorization rules.

Then, I defined two "factorization" rules; the *vice versa* function and the atomic negation function (Figure A.2B). The *vice versa* design is based on a promoter inversion element that switches from the computation of one function to another in response to an input. The atomic negation corresponds to the negation of each term of a function; this function is implementable

based on either promoter or terminator inversion-based elements by switching the orientation of the full cassette.

A.2 Implementation in Python of a set of factorisation rules using a brute force approach

I tried to implement these design rules in a Python script, starting from the design described in Chapter 2 based on the conjunction of AND and NOR functions.

The main difficulty of this work was to write the logic function in a minimized form to obtain the minimized design. As I did not know how to handle this logic problem, I developed a brute-force algorithm searching for specific patterns in the input logic truth table corresponding to identified biological modules, such as OR, NAND, XOR, and NXOR modules. I researched these patterns in various orders, and by composition with other patterns, tried to obtain the minimized logic form corresponding to the minimized circuit. I also included, after the search for a pattern, a search for *vice-versa* or atomic negation factorization. For each input truth table, I obtained a design for each order of the research of logic patterns. Therefore, I selected designs that permit the implementation of the logic truth table in a reduced number of strains.

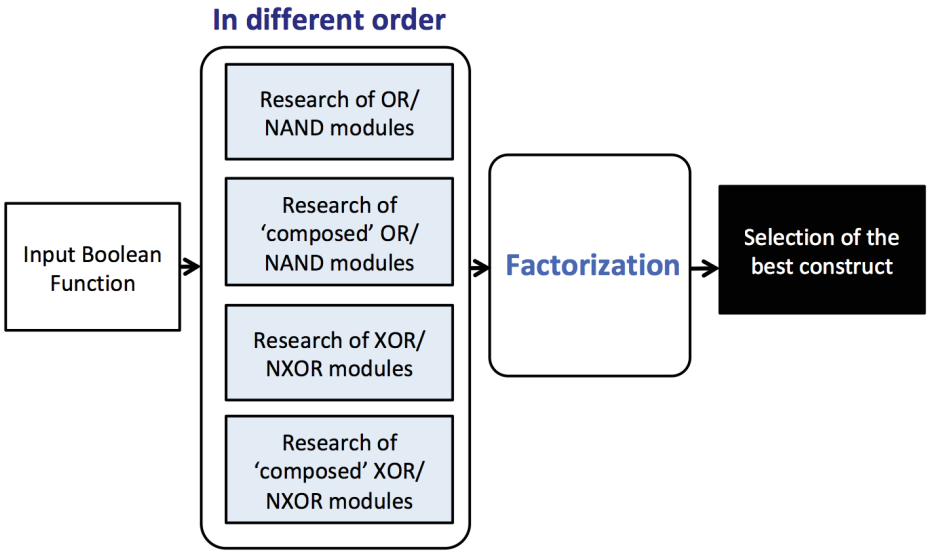


Figure A.3: **Algorithm of design.**

Using this algorithm, I generated the biological design of each 3-input Boolean function and for 10,000 4-input Boolean functions chosen randomly (Figure A.4). The time for computation of this algorithm is significant, as it is trying all possible minimization forms to obtain the simplest one. Thus instead of testing all functions, I chose to generate a subset of 4-input Boolean functions randomly obtained, which should represent the implementation of all 4-input Boolean functions.

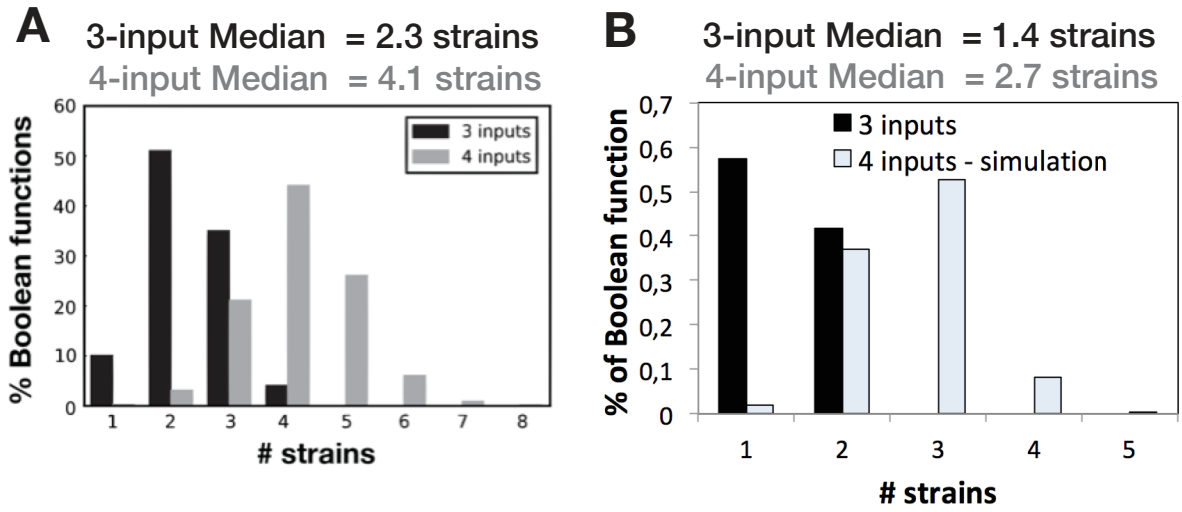


Figure A.4: **Number of strains required for the implementation of 3- and 4-input Boolean function using the design framework of Chapter 2 and the one detail here.** (A) Histogram for the design framework of Chapter 2. (B) Historgram for the design detail here. For 4-input only a sample of all 4-input Boolean designs have been generated to obtain this distribution.

Based on this algorithm, all 3-input Boolean functions are implementable within two strains, while four strains were required with the design from Chapter 2. For four inputs, the maximum number of strains required is five for these reduced set of functions, as compared to eight for the design of Chapter 2 (Figure A.4).

A.3 Discussion

Here I defined a reduced number of elements, modules, and composition rules. These rules permit the minimization of the implementation of recombinase logic in living organisms. I developed an algorithm for the systematic design of logic circuits based on these rules. However, the algorithm does not perform well, and the use of an efficient algorithm for simplification of logic functions in a factorized manner will be required. Nevertheless, based on this algorithm, I showed that this reduced set of rules permits the implementation of all 3-input logic functions within two strains.

In comparison to the generation process from Chapter 4, this systematic design strategy permits the design of logic devices following some biological constraints. Moreover, this principle is scalable to more inputs, as the rules are scalable to N-inputs and the design is straigthfoward.

This work has to be pushed further by (i) the development of an efficient simplification algorithm and (ii) the definition of additional design rules. Indeed, from the database generated in Chapter 4, it is probably possible to determine new design rules. For example, elements based on gene could be used. In addition, to be a fundamental logic problem, the development

of systematized design rules will permit the design of logic circuits integrating a high number of inputs.

Annex B

Supplementary Information - Introduction

B.1 Implementation of computation by regulation of transcription using Zinc Fingers, TAL effectors, and CRISPR.

In the main text, I mainly described the implementation of computation via regulation of transcription based on repressors. Zinc fingers, TAL effectors, and CRISPR can be also used for the implementation of these systems following the same principle than for repressors. I detail here the different tools and the logic circuits implemented using these tools.

Zinc Fingers (Cys2 - His2 ZFs) are small protein domains capable of binding to specific DNA sequences with high affinities. Their name comes from a finger-like structure motif which is stabilized via a common zinc ion. Zinc-finger DNA binding domains were engineered to recognize new nucleotide DNA sequences [Maeder 2008]. By fusing Zinc-finger domains to transcriptional activation or repression domains, VP64 and KRAB, Lohmuller and colleagues engineered 15 transcriptional activators and 15 transcriptional repressors in mammalian cells [Lohmueller 2012]. Employing hybrid promoters (for OR and NOR) and split intein mediated protein splicing (for AND and NAND), they constructed OR, NOR, AND, and NAND logic gates in mammalian cells.

TAL (transcriptional activator-like) effectors orgiginated from *Xanthomonas oryzae* bacteria and secreted when the bacteria infects a plant. TAL effectors can also bind specific promoter sequences and activate transcription [Boch 2010]. Their binding specificity is characterized by a simple correspondence between the amino acids in the TAL effector and the DNA bases of the target site. Artificial TAL effectors can then be easily designed to recognize specific DNA sequences. Two- and three-input TALE-based AND logic gates were engineered in embryonic stem cells using a split-intein protein-splicing strategy [Lienert 2013]. Therefore, the presence of two or three inputs were required to have an active TALE protein activating the output gene expression, which permits implementation of two- and three-input logic gates. Using TAL effectors, Gaber and colleagues built NOT and NOR gates [Gaber 2014] similar to TetR-family repressor gates. By positioning the DNA binding sites of the TAL effectors upstream

of the constitutive mammalian promoter (CMV), the expression of a TAL effector induces the repression of the output gene. This design permits implementation of NOT gates using one binding site and NOR gates using several binding sites upstream of CMV. As TAL effectors can serve as both the input and output of the gates, multiple TALE-based logic gates were connected to implement all 16 2-input logic functions in mammalian cells.

Finally, CRISPR-dCas9 was recently used to implement logic circuits in various organisms, using the binding specificity of dCas9 to sgRNA [Jusiak 2016]. Following a NOR-based design, various logic gates were built using sgRNA coupled with constitutive expression of dCas9 in *E. coli* [Nielsen 2014, Kiani 2014, Gander 2017]. sgRNA induces expression of a promoter composed of an operon that is complement to the sgRNA and tandem promoters for NOR gates are constructed based on multiple sgRNA:DNA interactions. Similar strategy was used in mammalian cells ([Nielsen 2014, Kiani 2014, Gander 2017]) and in yeast ([Nielsen 2014, Kiani 2014, Gander 2017]). In yeast, the Mxi1 domain was fused to dCas9, and it repressed gene expression in eukaryotic cells. Gander and colleagues used up to 7 orthogonal sgRNA:DNA pairs and 5 NOR gates in one circuit, implementing the majority of 2-input logic gates. The use of sgRNA:DNA pairs coupled with dCas9 permits the creation of up to 107 orthogonal NOT gates. Furthermore, CRISPR-dCas9-sgRNA toolbox was extend via the engineering of extending guide RNAs which included effector protein recruitment sites [Zalatan 2015].

B.2 *In vivo* implementation of sequential logic systems.

Most electronic circuits are based on a combination of sequential and combinational circuits. Sequential logic circuit behavior depends not only on the present value of the signal but also on the sequence of past inputs. In others words, sequential logic circuits produce history-dependent responses. We considered two different types of sequential logic circuits: dynamic sequential circuits and history-dependent circuits.

B.2.1 Circuits using rewritable memory devices.

Examples of dynamic sequential circuits include push on/push off circuits and flip-flop circuits. These circuits are sequential, as the current state of the circuits is dependent on both its previous state and on the state of the inputs, and dynamic as it can switch back to previous states.

Various dynamic sequential circuits have been implemented in living organisms, all circuits are based on feedback loops. First, the genetic toggle switch of Gardner and colleagues based on repressors is a dynamic sequential bistable circuit. It is based on the mutual inhibitory of two repressors responding to the two inputs. Various circuits have been designed based on similar designs. Basu and colleagues implemented a spatio-temporal dynamic circuit [Basu 2004]

based on repressors and cell-cell communication. Ajo-Franklin and colleagues implemented an autoregulatory transcriptional positive feedback in mammalian cells based on the LexA activator. Lou and colleagues implemented a push-on and push-off circuit based on repressors and feedback loops. Similarly to combinational logic, Urrios and colleagues implemented a memory device based on multicellular distributed computation [Urrios 2016]. This circuit is based on a double-negative feedback motif between two cells. Moreover, Urrios and colleagues built a feed-forward loop based on the same design strategy to implement a single pulse behavior [Urrios 2018].

A resettable recombinase-based circuit was engineered in *E. coli* [Bonnet 2012]. Recombinase-based logic circuits are by definition memory devices as with the single used property of integrase the system is irreversible. By using, Recombination Directionality Factor (RDF) to reset the circuit state, a recombinase-based toggle switch was implemented. More complex designs based on the association of recombinase and RDF were proposed by Subsoontorn [Subsoontorn 2012a, Subsoontorn 2014].

B.2.2 Irreversible history-dependent circuits.

History-dependent circuits that are not dynamic have been also implemented, such as systems to track cell lineage by random genome editing or to track the order of occurrence of inputs, such as history-dependent gene-expression programs.

Shipman and colleagues used CRISPR to randomly edit DNA [Shipman 2016]. By sequencing the genome, the cell lineage is re-traceable, making it a powerful tool to study development. Similarly based on CRISPR and random DNA editing, Tang and Liu implemented a rewritable multi-event analog recording circuit in bacteria and mammalian cells [Tang 2018]. Their circuit permits one to re-trace the occurrence of two inputs by genome sequencing and analysis.

Recombinase permits a more compact implementation of sequential circuits, as by default memory devices are implemented. The output is dependent on the presence of the input at any given time in the history of the system. Therefore, these circuits can be considered sequential. As the output is not dependent on the order of occurrence of the inputs, we called them asynchronous combinational logic circuits.

Based on recombinases, large history-dependent circuits have been implemented. Roquet and colleagues engineered a register of the order-of-occurrence of events by interdigitation of integrase sites. The state of the system, such as the order-of-occurrence of inputs, is readable by sequencing. Adding promoter(s), terminator(s), and output gene(s) between sites, history-dependent gene-expression programs from up to 3-input were implemented in *E. coli* [Hsiao 2016] [Roquet 2016]. These circuits as recombinase-based Boolean logic circuits are not resettable.

Annex C

A part toolbox to tune genetic expression in *B. subtilis*

During my master internship and the beginning of my thesis, I worked on the engineering of a part toolbox to tune genetic expression in *B. subtilis*. This work was in collaboration with Matthieu Jules and Vincent Suaveplane from INRA Jouy en Jousas. Caroline Clerté and Nathalie Declerk guided me for the experiments with the 2-photon microscopy and analysis. Hung-Ju Chang worked on the enginnering of the SsrA-tag toolbox. Jerome Bonnet and I designed the project and wrote the paper.

The following is the full paper and the supplementary data.

Published online 8 July 2016

Nucleic Acids Research, 2016, Vol. 44, No. 15 ***7495–7508***
doi: 10.1093/nar/gkw624

A part toolbox to tune genetic expression in *Bacillus subtilis*

Sarah Guiziou[1], Vincent Sauveplane[2], Hung-Ju Chang[1], Caroline Clerté[1], Nathalie Declerck[1], Matthieu Jules[2] and Jerome Bonnet[1,*]

[1]Centre de Biochimie Structurale, INSERM U1054, CNRS UMR5048, University of Montpellier, France and [2]Micalis Institute, INRA, AgroParisTech, Université Paris-Saclay, 78350 Jouy-en-Josas, France

Received May 25, 2016; Revised July 01, 2016; Accepted July 04, 2016

ABSTRACT

Libraries of well-characterised components regulating gene expression levels are essential to many synthetic biology applications. While widely available for the Gram-negative model bacterium *Escherichia coli*, such libraries are lacking for the Gram-positive model *Bacillus subtilis*, a key organism for basic research and biotechnological applications. Here, we engineered a genetic toolbox comprising libraries of promoters, Ribosome Binding Sites (RBS), and protein degradation tags to precisely tune gene expression in *B. subtilis*. We first designed a modular Expression Operating Unit (EOU) facilitating parts assembly and modifications and providing a standard genetic context for gene circuits implementation. We then selected native, constitutive promoters of *B. subtilis* and efficient RBS sequences from which we engineered three promoters and three RBS sequence libraries exhibiting ∼14 000-fold dynamic range in gene expression levels. We also designed a collection of SsrA proteolysis tags of variable strength. Finally, by using fluorescence fluctuation methods coupled with two-photon microscopy, we quantified the absolute concentration of GFP in a subset of strains from the library. Our complete promoters and RBS sequences library comprising over 135 constructs enables tuning of GFP concentration over five orders of magnitude, from 0.05 to 700 μM. This toolbox of regulatory components will support many research and engineering applications in *B. subtilis*.

INTRODUCTION

Synthetic biology aims at the rational engineering of novel biological functions and systems (1). By facilitating the engineering of living organisms, synthetic biology promise to enable the development of many new applications for health, manufacturing, or the environment. For example, in the past decade researchers have achieved complete synthesis of many compounds of interest in microorganisms, including several pharmaceuticals (2–4). Synthetic gene circuits enabling cells to perform tuneable oscillations (5), data storage (6–9), Boolean logic (10,11) and pattern formation (12) have also been engineered. Many genetic circuits have been developed in mammalian cells for diagnosis, disease classification and treatment (13–15). More recently, bacteria have been re-programmed to record inputs within the mammalian gut (16), detect metastases *in vivo* (17), or diagnose diabetes in human clinical samples (18).

These achievements rely on gene circuits of increasing size and complexity, and biological engineers had to finely adjust the expression level of many different genes at a time. For example, yeast-based synthesis of tebaine and hydrocone required the concerted production of up to 23 different enzymes (4). Refactoring heterologous nitrogen-fixation cluster or injectisome into *Escherichia coli* necessitated the coordinated expression of respectively 20 and 27 genes within a single bacterial strain (19,20). In this context, the availability of multiple regulatory components enabling fine-tuning of gene expression has become of utmost importance. In response to these needs, several libraries of components have been produced to regulate gene expression at several levels (mainly transcription and translation) for many organisms of interest including *E. coli*, *Saccharomyces cerevisiae*, and mammalian cells (21–24).

Many synthetic biology research and applications have been developed in bacteria using the Gram-negative model, *E. coli*, because of its ease of use and great numbers of regulatory components available. On the opposite, and despite overwhelming potential interests, the use of the bacterial Gram-positive model, *Bacillus subtilis,* has so far been limited.

Bacillus subtilis is a soil bacterium from the Firmicute phylum, which has been a long-time model organism (25). Complete genome sequence, along with transcriptome and proteome wide responses to various environmental conditions have been determined (26,27). Because it presents simple differentiation pathways, *B. subtilis* has been a model

*To whom correspondence should be addressed. Tel: +33 467417713; Fax: +33 467417913; Email: jerome.bonnet@inserm.fr

7496 Nucleic Acids Research, 2016, Vol. 44, No. 15

system for studying cell-fate decisions during development (28,29). For example, the role of stochastic fluctuation in gene expression during differentiation was recently probed in *B. subtilis* (30). Much of our understanding of bacterial chromosomal replication also comes from studies performed in this organism (31).

In addition to its role in basic research, *B. subtilis* is a biotechnology workhorse, being routinely used for the production of enzymes, antibiotics, but also for bioremediation (32–33). Indeed, from an engineering perspective, *B. subtilis* presents many advantages like natural competency, easy chromosomal integration, and an endogenous secretion pathway widely used in industrial protein production. The sporulation capacity of this bacterium facilitates storage conditions and spores can also be used as a convenient format for the surface display of many biomolecules (34). Finally, *B. subtilis* is non-pathogenic, has been classified by the U.S. Food and Drug Administration as a 'Generally Regarded As Safe' and was granted Qualified Presumption of Safety status by the European Food Safety Authority.

One reason for the modest usage of *B. subtilis* in synthetic biology is the lack of well-characterised, publicly available collections of regulatory elements to precisely tune gene expression levels in this organism. Recently, a collection of standardised components containing three constitutive promoters, two inducible promoters, five integration vectors, and few epitope tags has been produced (35). However, and despite its usefulness, the tunability range and the part diversity of this toolbox are still limited as compared with tools currently available for *E. coli*.

Here we engineered a toolbox of promoters, RBSs, and proteolysis tags to control expression of a gene of interest at the levels of transcription, translation and protein degradation in *B. subtilis* over many orders of magnitude (Figure 1A). We also standardised our measurement processes and characterised their robustness between two different laboratories using a newly defined reference construct. Finally, by using fluorescence fluctuation methods coupled with two-photon microscopy, we measured in living cells the absolute concentration of GFP produced by different members of our library. From this work we deliver a full part library enabling the tuning of GFP concentration from nanomolar to millimolar concentrations (15 to 270 000 GFP molecules/cell, respectively). This extensive parts library enabling precise tuning of gene expression will be useful for the broad research and engineering community working with *B. subtilis*.

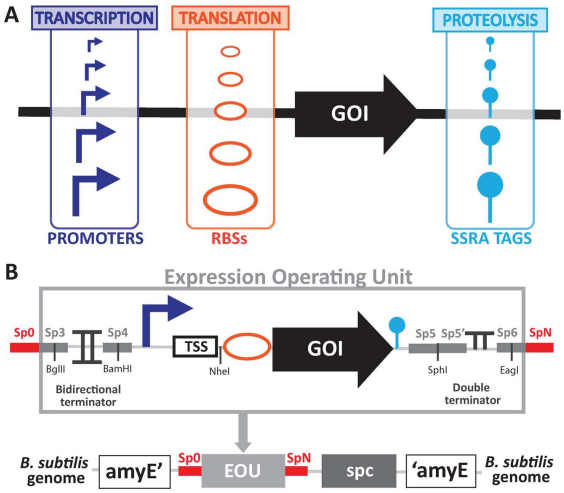


Figure 1. Design of a toolbox to tune genetic expression in *Bacillus subtilis*. (**A**) We engineered libraries of regulatory components with different strengths and sequences to tune genetic expression of a gene of interest (GOI): constitutive promoters to tune transcriptional efficiency, RBSs to tune translational efficiency and degradation tags to tune proteolysis rate of the protein of interest. (**B**) Architecture of our standardised and modular Expression Operating Unit (EOU). The EOU is composed of the standard regulatory elements (promoter, RBS, GOI, degradation tag), a standardised sequence of 8 nucleotides at the TSS position, a bidirectional terminator and a double terminator to insulate the cassette from genetic context. Spacers (SpX) of 40 bp designed to facilitate one-step isothermal assembly as well as several restriction sites enable simple construction and switching of parts. The EOU is integrated in the *B. subtilis* genome by double-crossover at the *amyE* locus (alpha-amylase gene). The EOU is coupled with a cassette coding for the spectinomycin adenyltransferase (*spc*) to allow antibiotic selection of the integrants.

MATERIALS AND METHODS

B. subtilis transformation and chromosomal integration

Bacillus subtilis strains derived from BSB168, a *trp*+ derivative of *B. subtilis* 168 (26,27). *B. subtilis* strains were grown on either LB media, M9 minimal media supplemented with glucose and malate (0.5% glucose and 0.3% malate) (M9-MG) or CHG medium supplemented with glucose (0.5%) (www.basysbio.eu) (CHG). Complete protocols and media composition for competent cells preparation and chromosomal integration (adapted from (36)) can be found in supplementary materials.

Briefly, synthetic constructs were integrated using pDG1730 integration vector into the *amyE* locus of *B. subtilis* genome by double-crossover integration. Positive selection of integration was performed with spectinomycin at 100 μg ml^{-1} and negative selection of single crossover integration events with erythromycin at 0.5 μg ml^{-1}. Colony PCR for verifying part integrations were realised using Kapa 2G Robust PCR kit (Clinisciences, buffer B). The PCR products were then sequenced.

Molecular biology

We used pDG1730 (Genbank U46199,(37)) that we obtained from the *Bacillus* Genetic Stock Center (http://www.bgsc.org) as our backbone plasmid for *B. subtilis* integration into the *amyE* locus. All plasmids used in this study were derived from this vector and fragments assembled using one-step isothermal assembly (38) or restriction enzymes following standard molecular biology procedures. Restriction enzymes were purchased from New England BioLabs (NEB, Ipswich, MA, USA). PCR were performed using Q5 PCR master mix (NEB), primers and Gblocks were purchased from IDT (Louvain, Belgium; Carlsbad, USA). Plasmid extraction and DNA purification were performed using kits from Biosentec (Toulouse, France). Sequencing was realized by GATC Biotech (Cologne, Germany). All primers

Nucleic Acids Research, 2016, Vol. 44, No. 15 **7497**

sequences and details on molecular biology protocols are available in supplementary materials.

Construction of randomised libraries and integration in *B. subtilis*

The various promoters, RBS sequences, or degradation tags libraries were generated by amplifying the GFP gene using primers containing the regulatory region of interest degenerated at strategic positions. This PCR products library was then digested by specific restriction enzymes and cloned into our standard EOU accordingly digested.

PCR amplification and cloning. For the initial P_{veg} libraries (Supplementary Figure S2A, no standard TSS element), P_{veg} was randomised following three different strategies: randomisation of −10 and −35 boxes, randomisation of −35 box, or randomisation of the −10 box. For the P_{veg} libraries (with standard TSS element), only the −10 box was randomized at three positions. Vectors and amplified fragments were digested by *Age*I and *Sph*I and ligated. P_{serA} and P_{ymdA} were randomized following two strategies (randomisation of −10 box or randomisation of the region flanked by the −35 and −10 boxes). Vectors and amplified fragments were digested by BamHI and SphI and ligated. RBS and degradation tag libraries were generated following a similar procedure. Vectors and amplified fragments were digested by NheI and SphI.

Ligation and transformation into E. coli. Vectors and fragments were ligated using T4 ligase (NEB) at 16°C overnight. DNA was transformed in *E. coli* using electro-competent cells and plated in large selective agar plates (∼ 4 000 colonies per library). After overnight growth, all clones were scrapped from agar plates and grown at 30°C on 5 ml of LB during 2 h. 1 ml of culture was used for DNA extraction. Target sequence randomisation was verified by Sanger sequencing.

Batch integration into B. subtilis. For batch integration of libraries in *B. subtilis*, the integration protocol was performed using 10 μg of variant DNA in 10 mL of *B. subtilis* competent cells. At the end of integration protocol, two aliquots of 500 μl of cell cultures were plated on spectinomycin or erythromycin agar plates for quantification of integration efficiency (∼100 double-crossover events for 500 μl of competent cells, hence 2 000 clones per libraries using batch integration). The remaining cells were centrifuged at 1 600 g for 10 min, the supernatant was removed, cells were re-suspended in 10 ml of spectinomycin LB and grown 16 h at 30°C to avoid elimination of slowly-growing cells (39), before being either sorted by FACS or conserved in glycerol stocks.

Fluorescence activated cell sorting of libraries

For each library, glycerol stocks of *B. subtilis* variants were inoculated in 5 ml LB and grown 16 h at 30°C. The next day, cells were diluted and grown on M9-MG. Then, cells from the libraries were sorted using a S3 Cell Sorter (Biorad). The expression level range was divided in seven different regions, or bins, in which cells were sorted according to their GFP expression level. 10 000 bacteria were sorted into each bin and were plated on selective agar plates. For each promoter library, four variants per bin were selected for further characterisation, for a total of 28 variants per library. For each RBS library, 20 variants per bin were selected for a total of 140 variants characterised. All variants were entirely sequence verified. We excluded clones containing unexpected mutations (e.g. within the GFP sequence or the RBS for promoter libraries) and chose the variants presenting the lowest dispersion around the median value of the fluorescence intensities, and the lowest variability in gene expression between experiments performed on different days.

B. subtilis cell culture for parts characterisation

For measurements performed on exponential phase, 96 deep well plates filled with 1 ml of LB per well were inoculated with clones from fresh streaked plates. Plates were grown 16 h at 37°C. Cultures were diluted 40 times on 200 μl of LB in 96-well plates and grown 2 h. Then, cultures were diluted 40 times on 200 μl of M9-MG and grown at 37°C until OD reached ∼0.3–0.4 (∼3 h). Cultures were diluted 40 times on 200 μl of M9-MG and cells were immediately analysed on the flow-cytometer. For measurements performed on stationary phase, 96 deep wells plate filled with 1 ml of LB per wells were inoculated with clones from fresh streaked plates. Plates were grown 16 h at 37°C. Cultures were diluted 40 times on M9-MG and measure on flow-cytometer within the hour.

Flow-cytometer measurements and analysis

Quantification of expression levels of all strains were performed using Attune NxT flow-cytometer (Thermofisher) equipped with an autosampler. Experiments were performed on 96 wells plates with three replicates per plates. In each plate, the reference constructs and the negative control strain (integration of pDG1730 without EOU) were present. For a given part, each measurement procedure was performed in triplicates on three different days.

For flow cytometry measurements, 10 000 bacteria events were analysed. A gate was previously designed based on forward and side scatter graphs to remove debris or spores from the analysis. GFP fluorescence intensity was measured using excitation by a 488 nm laser and a 510/10 nm filter (BL1). mKate2 excitation was performed by a 561 nm laser and filter 615/25 nm (YL2). Voltages used were FFS: 440, SSC: 340, BL1: 490, YL2: 620.

Data were analysed using the Attune NxT software. Flow-Jo (Tristar) was used for data representation. Statistical values for each channels of the sample were calculated and exported. For each independent experiment, the median fluorescence intensity of the bacterial population for each replicate was extracted. Then, the mean fluorescence intensity was calculated from the three replicates. The mean values and standard deviation from three independent experiments were then calculated. Relative expression units were calculated for each independent experiment by dividing the mean fluorescence intensities values measured from the synthetic constructs by the mean fluorescence intensity

7498 Nucleic Acids Research, 2016, Vol. 44, No. 15

measured from the reference construct. All raw data files are available in supplementary materials.

Plate reader experiments and analysis

Quantification of expression levels of promoters was performed using a BioTek Cytation 3 in Montpellier (France) and a BioTek Synergy II in Jouy-en-Josas (France). Experiments were performed using 96-well plates with three replicates per plate, and in each plate was always cultured the reference construct and the negative control strain. Three independent experiments were performed. To begin, 96 deep-well plate filled with 1 ml of LB per wells were inoculated with clones from fresh streaked plates. Cells were grown for 16 h at 37°C. Cultures were then diluted 400 times on 200 μl of LB in 96-well plates and grown until OD reached ∼0.3–0.4. At this point, cultures were diluted 400 times on 200 μl of CHG and grown at 37°C until OD reached 0.3–0.4. Cultures were diluted 400 times on 200 μl of CHG and grown for 16 h on plate reader with measure of green fluorescence intensity (ex. 485/20 nm, em. 528/20 nm) and absorbance (at 600 nm) every 10 min. Absorbance at 900 nm and 977 nm (Abs_{900}, Abs_{977}) were read once at the beginning of each experiment in order to correct the OD_{600} to an optical path length of 1 cm using the following equation: $(Abs_{977} - Abs_{900})/0.18$ (40). Fluorescein was present on the microtiterplate at two different concentrations (1 and 10 nM) in duplicates. Each culture was performed in triplicates. Polynomial and exponential functions were used to fit the experimental datasets of GFP and biomass, respectively (26), and to deduce the rates of biomass and GFP productions along the growth. GFP concentration was estimated as GFP per OD_{600}, ($\frac{GFP}{OD}$), at each time point. In steady-state growth (μ = constant), $\frac{GFP}{OD}$ is constant. GFP concentration (also referred to as activity) was expressed in Relative Expression Units (REU) using our reference construct. Data were analysed using custom Matlab scripts.

2-photon fluorescence microscopy experiments and number and brightness (N&B) analysis

Cells were cultivated in 24-well microplates in 1.5 ml M9-MG and maintained in exponential phase by dilution for at least 16 h to avoid the presence of spore. Aliquots of cell cultures were removed to perform simultaneously microscopy and flow-cytometry measurements. For microscopy experiments, 1 ml of culture at OD_{600} ∼0.2–0.5 was centrifuged at 1 600 g for 2 min, the supernatant was removed, and the cell pellet was re-suspended in M9-MG medium to a final OD_{600} of ∼25. A 2.5 μl aliquot was placed on a 2% agarose–M9 pad and cells were imaged using an Axiovert 200M inverted microscope (Zeiss, Germany) equipped with an ISS laser scanning module and an ISS Alba (ISS, Champaign, IL, USA) with two-channel APD detection (see (41) for details). Each experimental day, the laser was re-aligned and the 2-photon excitation volume was calibrated using a standard fluorescein solution. We used a laser power of 6 mW for all experiments, and an excitation wavelength of 930 nm for GFP. We measured our excitation volume to be of 0.07 fl, about a seventh of the bacterial cell volume. For each strain, four different fields of view (FOV) were imaged (256 × 256 pixels, 30 μm × 30 μm), each containing about 200–300 individual cells. For each FOV, a series of 50 raster-scanned images were recorded using a 40 μs laser dwell time per pixel. The negative control (NC) strain (expressing no GFP) was cultivated and imaged under identical conditions to determine the autofluorescence background level for each experimental day.

A summary of the procedure used for number and brightness (N&B) analysis derived from (42) is given below and detailed explanation of the method adapted for bacterial cells can be found in (41). Individual cells in each FOV were contoured automatically with manual correction using the Patrack software (43). For each FOV, fluorescence fluctuations (δF) from the average intensity over 50 scans ($<F>$) were first calculated at each pixel, providing pixel-based maps of the true (shot noise corrected) molecular brightness of the diffusing fluorescent particles, ϵ:

$$\varepsilon\ (x,\ y) = \frac{\delta F^2\,(x,\ y) - \langle F\rangle\,(x,\ y)}{\langle F\rangle\,(x,\ y)}$$

For each FOV, the average molecular brightness ϵ_{FOV} was determined using only the M pixels encompassed within all the cells of the FOV, and the number of fluorescent particles detected in the excitation volume within each cell was calculated:

$$n_{\text{cell}} = \frac{1}{M}\sum_{j=1}^{M}\frac{\langle F\rangle^2\,(j)}{\delta F^2\,(j)}\frac{\varepsilon_{\text{FOV}}+1}{\varepsilon_{\text{FOV}}}$$

The molecular brightness of GFP (ϵ_{GFP}) was estimated for each experimental day by averaging ϵ_{FOV} measured for strains expressing moderate amount of GFP (i.e. $40 < <n_{cell}> < 400$). For each strain including the background strain, the average number of GFP equivalent molecules detected in the intracellular excitation volume was calculated using the daily ϵ_{GFP} value:

$$\langle N\rangle = \frac{\langle n_{\text{cell}}\rangle\ \langle\varepsilon_{\text{FOV}}\rangle}{\varepsilon_{\text{GFP}}}$$

The average intracellular concentration of GFP molecules $<N_{GFP}>$ corrected for the auto-fluorescence background was obtained by subtracting to $<N>$ the average number of GFP equivalent molecules calculated for the NC strain ($<N_{nc}>$) and dividing by the excitation volume inside cell ($vol_{ex} = 0.07$ fL, (41)) and the Avogadro number (N_A) :

$$\langle N_{GFP}\rangle = \frac{\langle N\rangle - \langle N_{nc}\rangle}{vol_{ex}\ N_A}$$

The average number of GFP molecules per cell can be estimated by multiplying $<N_{GFP}>$ by N_A and the average cell volume. Under our experimental conditions, we estimated the average cell volume to be a 0.5 fl ± 0.2, which was calculated from several images obtained for different strains and experimental days.

For the seven constructs measured using 2p sN&B methods, we obtained a linear correlation between concentration of GFP and fluorescence intensity measured using flow-cytometer in arbitrary unit. We assumed that the correlation

Nucleic Acids Research, 2016, Vol. 44, No. 15 **7499**

is linear in the full expression range. As one REU corresponded approximately to 7.21×10^4 AU (fluorescence Arbitrary Unit) and intracellular GFP concentration is equal to 0 for the background construct, we obtained a correlation between intracellular GFP concentration and relative promoter unit corrected by the background of 1 ($REU - REU_{NC}$) = 144 ± 24% [GFP] (μM). To determine the correlation error, a correlation was determined individually for each seven values, and the 24% error corresponded to the highest error between the various correlations.

RESULTS

Design of a standard and modular Expression Operating Unit (EOU) for *Bacillus subtilis*

Our first goal was to design a genetic architecture supporting rapid, simple, and reliable parts assembly or exchange. An additional specification was to provide a standard genetic context for gene circuits characterisation. We thus designed a standardised and modular Expression Operating Unit (EOU, (22)) for controlling gene expression (Figure 1B). Since chromosomal integration is the general gene expression strategy used in *B. subtilis*, we placed our EOU into the pDG1730 vector (37), which is used for targeted chromosomal integration at the *amyE* locus. The basic EOU contains a gene of interest (GOI), which can be flanked by various regulatory components: a promoter, a ribosome binding site, and possibly a degradation tag. We also designed an Expression Operating Unit architecture for expression of two genes and for inducible gene expression (Supplementary Figure S1). We placed transcriptional terminators at both extremities of the EOU to stop transcription and to insulate the constructions from transcription incoming from neighbouring regions.

We also tried to avoid context effects due to the Transcription Start Site (TSS) region. In fact, at some promoters, the RNA polymerase can initiates transcription at two or three alternative neighbouring bases, +1, +2 or +3 (as illustrated in *E. coli* (44) and in *B. subtilis* by (45)). The probability to start transcription at +1, +2 or +3 most likely depends on the nature of the nucleotides present at these positions and on the intracellular level of the cognate NTPs (45,46). Unexpected context effect affecting transcription efficiency could therefore arise if we used various RBSs with different nucleotides compositions. Different 5′-untranslated regions could also affect gene expression levels by changing mRNA decay kinetics. We thus decided to standardise the (TSS) region of our constructs. We defined a standard TSS element (GGAGAAAA) corresponding to the first 8 nucleotides of the TSS of the P_{fbaA} gene (encoding the fructose-bisphosphate aldolase), and placed it between the promoter and the RBS.

We incorporated 40 bp spacers at several positions to facilitate assembly and switch of parts using one-step isothermal Gibson assembly (38). In addition, we placed various cutting sites for different restriction enzymes so that parts can also be exchanged by restriction/ligation reactions. To quantify gene expression, we used a Green Fluorescent Protein (GFP) as a reporter. Based on previous work in *B. subtilis* (47), we selected the superfolder GFP (sfGFP(sp), simply named GFP from here and below), which is very efficiently expressed in *B. subtilis*.

Definition of a reference construct

The use of reference objects facilitates measurements reproducibility and design, and has a long-standing history of success in various engineering fields. In synthetic biology, a reference construct (using promoter BBa_J23101 coupled with GFP) has been used in *E. coli* as an *in vivo* standard facilitating comparison of *in vivo* promoter activity measurements (48). Expression of parts activity in Relative Expression Units (REU) using this reference construct allows reduction of data variation due to difference in day-to-day and lab-to-lab test conditions and set-ups. Previously, a reference construct had been proposed as well for *B. subtilis* (35). However, we found that the activity of this construct was too low to serve as a reliable reference for characterizing expression levels over a wide dynamic range (i.e. a small experimental variation from the reference construct greatly affects the calculated REU of all characterised constructs). We therefore designed a new reference construct for *B. subtilis*.

To this aim, we prepared a first library of randomised promoters based on the promoter P_{veg}, well-known to be constitutive (35,49, Supplementary Figure S2). From this library spanning 3 orders of magnitude in GFP expression levels, we selected a reference promoter (P_{REF}) exhibiting an intermediate expression level. The full reference construct is composed of the P_{REF} promoter sequence, a strong RBS sequence (named RBS R0) typically used with the *B. subtilis* IPTG inducible promoter $P_{hyperspank}$ and the GFP coding sequence. This reference construct was used in all subsequent experiments to express gene expression as Relative Expression Units (REU) instead of arbitrary fluorescence intensity units.

Choice and characterisation of ten *B. subtilis* constitutive promoters

We aimed at designing synthetic libraries of constitutive promoters spanning a wide dynamic range of transcriptional efficiencies in *B. subtilis*. Such constitutive promoter libraries are essential tools for precise engineering of genetic circuits. For example, in metabolic engineering, the expression level of the different enzymes of the pathway has to be precisely tuned (50). In order to identify a first set of natural constitutive promoters from the *B. subtilis* genome, we used data recently produced by the BaSysBio consortium (27). This consortium mapped the transcriptional architecture, metabolic and networks behaviour of *B. subtilis* at a large scale and over 100 different conditions.

We searched the mRNA expression database (http://genome.jouy.inra.fr/cgi-bin/seb/index.py) for genes which transcript levels were relatively constant over the full range of experimental conditions. We chose ten genes with promoter regions known or predicted to be dependent on the housekeeping sigma factor σ^A (Figure 2A). Two of these promoter regions, P_{veg} and P_{lepA}, had already been isolated and characterised in *B. subtilis* (35,49). All the other promoter sequences were arbitrarily defined as the 50 first nu-

7500 Nucleic Acids Research, 2016, Vol. 44, No. 15

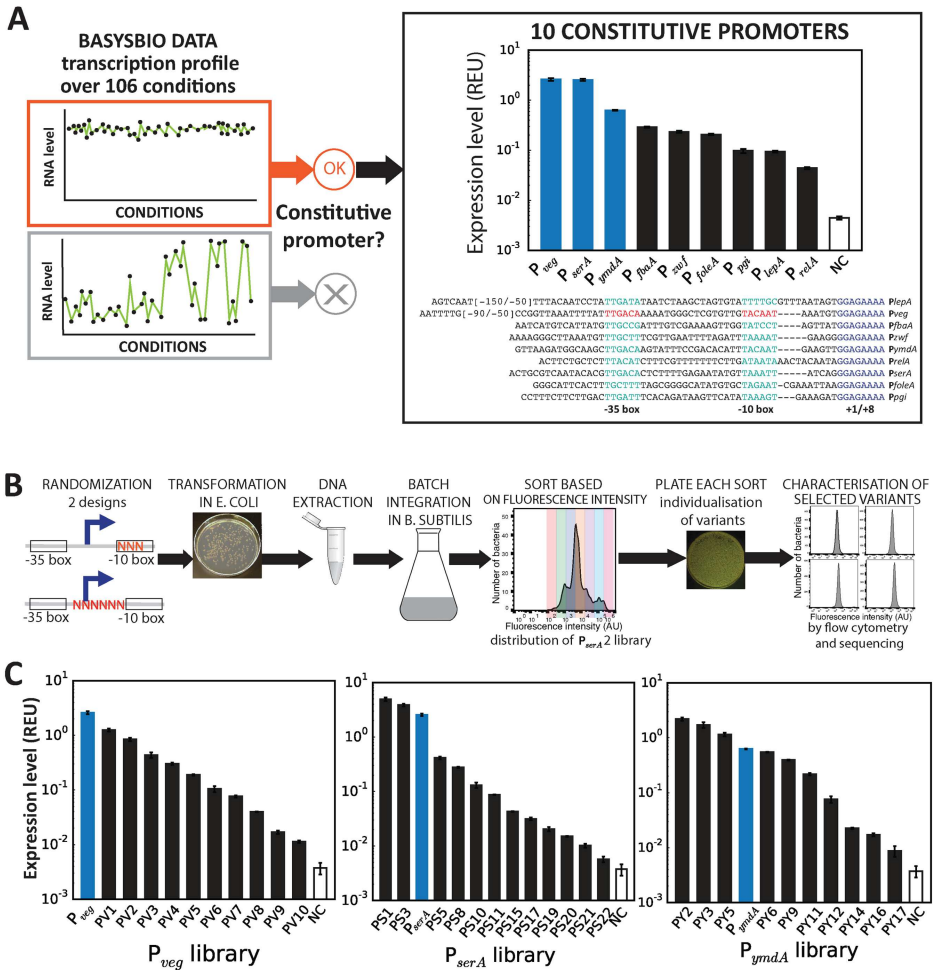


Figure 2. Engineering constitutive promoter libraries to tune transcription level in *B. subtilis*. (**A**) *B. subtilis* constitutive promoters were selected from the BaSysBio database based on a regular transcription profile under 104 different conditions. Two sketches of potential transcriptional profiles are depicted, in which the y-axis correspond to the mRNA levels and the x-axis correspond to different conditions. Upper panel: a constitutive gene active in most of the conditions, and thus displaying a desirable profile for constitutive promoter library design. Lower panel: a gene showing significant variation in expression over the different conditions and therefore not a suitable candidate. Based on this framework, 10 constitutive promoters were selected and characterised using our standardised cassette, using R0 as RBS and a superfolder GFP as a GOI. The cassette was integrated into the *amyE* locus of the *B. subtilis* genome. Expression levels were measured by flow-cytometry in exponential phase (see methods). Expression levels are expressed in Relative Expression Unit (REU) and error-bars represent the standard deviation over 3 independent experiments. Promoter sequences are represented with their TSS sequences highlighted in blue and their −10 or −35 box aligned and highlighted in red for experimentally validated sequences and in green for putative sequences. Full library measurements data are available in supplementary data files 1 and 2. (**B**) Workflow to engineer promoter libraries. (i) Randomisation of promoters by PCR using randomised oligonucleotides: two different designs are depicted; design 1: randomisation of three nucleotides in the −10 region; design 2: randomisation of six nucleotides between -35 and -10 regions. (ii) Cloning of the randomised fragments in a shuttle vector. (iii) After transformation in *E. coli*, extraction of plasmid DNA from the pool of transformed *E. coli*. (iv) Batch integration in *B. subtilis* of the extracted plasmid DNA by double-crossover at the *amy*E locus. (v) Sorting of the library based on fluorescence intensity into seven different bins to obtain various pools of variants within the same range of expression level. (vi) Plating of sorted cells onto selective agar to isolate individual variants. (vii) Characterisation of four variants per gates: flow cytometer measurements and sequencing of colony PCR products from the integrated constructs. (**C**) Three curated promoter libraries from three different parent sequences were obtained by following the process describe above. Expression levels are in relative expression units (REUs) and obtained by flow-cytometry measurements performed in exponential phase. Error bars: ±SD over three independent experiments. See methods and supplementary material for more details. Full library measurements are presented in Supplementary Figure S3 and supplementary data files 1 and 3.

Nucleic Acids Research, 2016, Vol. 44, No. 15 **7501**

cleotides upstream of the putative transcriptional start according to the alignment with the consensus sequence of σ^A recognition elements. Of note, when we added our standard TSS element to P_{veg} and to the reference promoter we observed a marked reduction in GFP expression (Supplementary Figure S3), confirming the influence of this region on transcriptional efficiency (44–46).

We introduced synthetic DNA fragments comprising the selected promoter regions in our standard EOU upstream of the standard TSS element, a strong RBS (R0), and the GFP coding sequence. The constructs were then integrated at the *amyE* chromosomal locus and GFP expression levels were measured by flow-cytometry in exponentially growing cells (Figure 2A). We observed high-level GFP production from the 10 selected promoters, demonstrating that the standard EOU we designed is a suitable reporter system for evaluating the relative transcriptional efficiency of promoter sequences integrated in the *B. subtilis* chromosome. The promoters activity went from 10- to 600-fold over the auto-fluorescence background level measured in the negative control (NC) strain. In all, the ten promoters spanned a 60-fold range in expression levels.

Construction and characterisation of promoter libraries

Recently constructed libraries of parts for *E. coli* or *S. cerevisiae* allow tuning of gene expression over a 10 000-fold range (21–23). In addition, if many parts are to be used in combination to engineer more complex gene circuits or pathways, different part sequences are required to avoid recombination due to high sequence similarity (51).

In order to increase the sequence diversity and expression dynamic range of our promoter parts, we randomized three different 'parent' promoter sequences. From our set of 10 constitutive promoters, we chose the three strongest: P_{veg}, P_{serA} and P_{ymdA}. All three promoters have a strong consensus signature for the *B. subtilis* household sigma factor σ^A (TTGACA(-35)-N14-tgnTATAAT(−10)) and we expected that randomisation would more likely result in a loss rather than in a gain of function.

We first randomized the P_{veg} promoter, targeting simultaneously or independently nucleotides within the −35 and −10 boxes (Supplementary Figure S2A). Randomization of 3 nucleotides in the −10 box gave satisfactory results and was thus applied to the P_{serA} and P_{ymdA} promoters. For these two promoters, we also tested a second randomization strategy, targeting six nucleotides (−21 to −16) in the spacer region between the −35 and −10 boxes (Figure 2B, Supplementary Figure S2B).

We cloned the randomized promoter sequences libraries into our standard gene EOU, using RBS R0 and GFP as reporter, amplified them in *E. coli* and integrated them within the *B. subtilis* genome. We then used Fluorescent Activated Cell Sorting (FACS) to isolate subpopulation of cells exhibiting specific transcriptional activity by sorting variants into seven different bins of varying GFP fluorescence intensity (Figure 2B). Then, for each bin, we characterised four variants using flow-cytometry (see materials and method for details). After screening, characterisation, and curation, we ended up with a set of 10–13 promoter variants for each library (excluding the wild-type sequence), chosen to span the highest magnitude in expression level (Figure 2C and Supplementary Figure S4).

For the first promoter library based on P_{veg}, randomization of the −10 box was sufficient to obtain a library covering a wide range of transcriptional activities (∼100 fold range, Figure 2C, left panel). However, for P_{ymdA} and P_{serA} libraries, randomization of the −10 box produced mostly promoter variants displaying no or very weak activity, and only a very few efficient promoters were identified (Supplementary Figure S2). In contrast, randomization of the spacer region between the −35 and −10 boxes generated mostly strong to medium promoters. Interestingly, some members of both the P_{serA} and P_{ymdA} libraries were more efficient than their parental promoter. By combining variants produced using both randomization strategies (see Supplementary Figure S4 for details), we obtained promoter libraries spanning a 900-fold range in REU (Figure 2C, middle and right panels).

Construction and characterisation of RBS libraries

Tuning gene expression at the level of translation can be essential depending on the gene circuits. For example, if a well-characterised inducible promoter is used, the simplest strategy to tune its expression dynamic range is to use a different ribosome-binding site (18,52). To tune translation efficiency, we first selected a set of 8 ribosome-binding sites derived from RBSs found in highly and constitutively expressed genes. The chosen RBS sequences comprise 20–24 bp and all of them but one (R4) contain the consensus (GGAGG) Shine-Dalgarno (SD) sequence for most bacterial species, including *B. subtilis* (53) albeit flanked by sequences of various compositions and lengths (Figure 3A). RBS sequence R0, which we used for screening our promoter libraries, is an optimized sequence typically used with the *B. subtilis* IPTG inducible promoter $P_{hyperspank}$. RBS sequences R3 and R5 to R7 are synthetic sequences derived from the RBS sequence of the strongly expressed glycolytic *fbaA* gene (52). R1 and R4 are the natural sequence of the putative RBS sequence from the *B. subtilis tufA* (R1) and *gltX* (R4) genes, encoding respectively the elongation factor TU and the glutamyl-tRNA synthetase. R2 is a synthetic RBS sequence designed to maximize binding of the ribosome by pairing with up to 15 nucleotides at the 3′ end of the *B. subtilis* 16S rRNA sequence (54). We characterised the activity of these 8 RBS sequences in the context of our standard EOU integrated at the *amyE* locus, using P_{veg} as promoter and GFP as reporter (Figure 3B).

We measured GFP production in *B. subtilis* cells in exponential phase and we observed high expression levels with all 8 RBS, from about 50-fold up to 600-fold above the background level (Figure 3C). Interestingly, the synthetic RBS R2 supposed to maximize ribosome binding is not the most efficient sequence, in agreement with a previous report (55). In order to tune translation efficiency over a large dynamic range, we engineered three libraries of RBS parts, starting with the three strongest ribosome binding sites, R0, R1 and R2 as parent sequences. We then performed PCR using degenerated oligonucleotides to randomize six nucleotides upstream the start codon and comprising the Shine-Dalgarno sequence (XGGAGG or GGAGGX), a

7502 Nucleic Acids Research, 2016, Vol. 44, No. 15

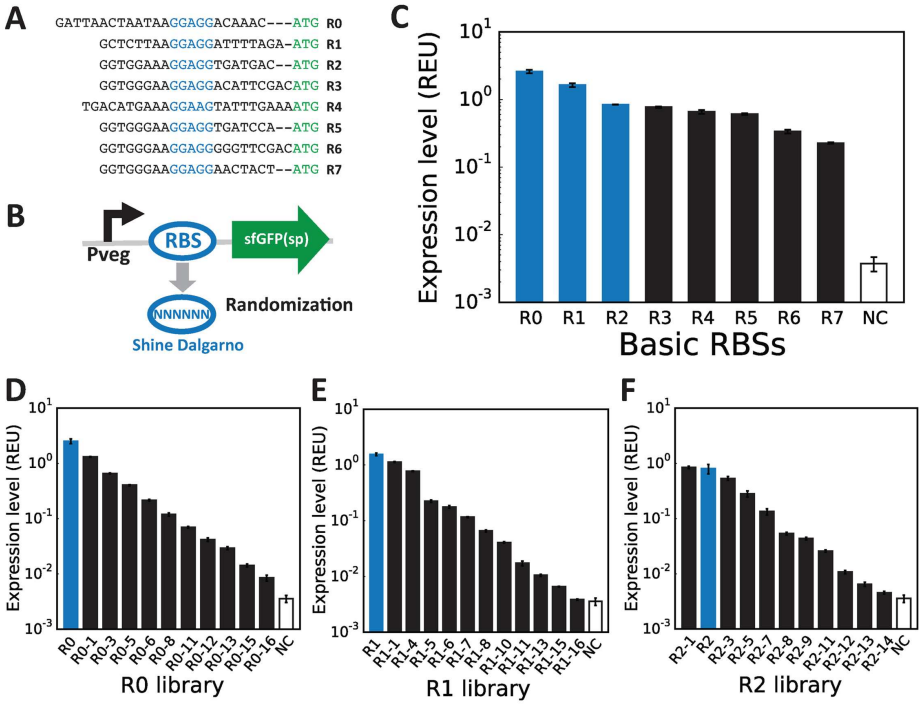


Figure 3. RBS libraries to tune translation level in *B. subtilis*. (**A**) A set of nine RBSs was selected. Their sequences are represented with their Shine Dalgarno sequences in blue and the start codon of the GOI in green. (**B**) RBSs were characterised using the standardised cassette, using P_{veg} as a promoter and the superfolder GFP as a GOI. The cassette was integrated into the *amy*E locus of the *B. subtilis* genome. To engineer RBS libraries, we randomised six nucleotides inside and around the Shine-Dalgarno of the RBS parent sequences. (**C**) Expression levels from strains containing the eight RBSs driving GFP expression in exponential phase measured by flow-cytometry. Error bars: ±SD over 3 independent experiments. (**D**–**F**) Three RBS sequence libraries from 3 different parent sequences: R0, R1 and R2 (variants are engineered using the workflow described in Figure 2B). Flow-cytometry data are from 3 independent experiments performed in triplicates. Error bars: ±SD over three independent experiments. Full library measurements are presented in Supplementary Figure S4 and supplementary data files 1 and 4.

well-known method to tune translation efficiency (11) (Figure 3B). When screening each of our three RBS mutant libraries, we found that most of the *B. subtilis* transformants displayed fluorescence intensity close to background level, indicating that most of the mutations led to not or poorly active RBS sequences (Supplementary Figure S5A). Nevertheless, by re-applying the same sorting strategy as for promoters while characterising more variants per bin (28 for RBSs versus 4 for promoter libraries), we obtained three libraries with different sequences each composed of 10–11 RBSs with translational activities spanning ∼800 fold range (in REU) (Figure 3D–F, Supplementary Figure S5B).

Because gene expression efficiencies are known to be affected by interactions between the 5′UTR and the gene of interest (GOI), we measured the activity of our initial set of eight RBSs coupled to the coding sequences of two different fluorescent proteins (Figure S6A): GFP and the red fluorescent protein mKate2 ((56), named RFP from here on). Both proteins present 45.9% identity within their first 100 nucleotides. By plotting REU values for RBS coupled to GFP or RFP we obtained a linear correlation fit with a coefficient of determination of ∼0.87 (Supplementary Figure S6B). However, two RBS sequences (R4 and R7) stood apart from the linear correlation curve: R7 appeared more efficient for RFP than for GFP expression whereas R4 was functional with GFP but not with RFP.

We tried to alleviate this putative context effect by using a bicistronic design (BCD), a system described in *E. coli* containing two concatenated SD sequences that reduces the influence of the GOI sequences on translation initiation efficiency (57). We designed BCDs containing R4 and R7 and coupled them with GFP or RFP expression units. We then measured GFP and RFP expression from these BCDs and their monocistronic counterparts (Supplementary Figure S6C and S6D). For R7-BCD, we observed an increase in GFP expression level compared to monocistronic R7, while RFP expression levels remained similar. For R4-BCD, GFP expression levels were reduced compared to monocistronic R4, whereas RFP expression was greatly improved by using the BCD. These results suggest that BCDs can also be used in *B. subtilis* to mitigate context effects arising from 5′UTR-GOI interactions. However, it is hard from this small number of data points to conclude on a general applicability of BCDs in *B. subtilis,* and deeper investigations are needed.

Nucleic Acids Research, 2016, Vol. 44, No. 15 **7503**

Engineering libraries of SsrA proteolysis tags for *B. subtilis*

While some genetic circuits only require controlling gene expression levels at the transcriptional or translational levels, others need an additional layer of control at the post-translational level. In particular, tuning of protein degradation rate is essential to the dynamic behaviour of some synthetic gene circuits like oscillators or rewritable data storage circuits using recombinases (5,9,58). In *E. coli*, active proteolysis can be triggered by using the SsrA system, in which a small 14 amino-acids peptide added to the C-terminus of a protein acts as a molecular barcode through which polypeptides are targeted for proteolysis by cellular proteases from the AAA+ family like ClpXP (Figure 4A). Modifications of the three last residues of the SsrA peptide were shown to alter the affinity of the peptide for the protease, enabling researchers to tune protein degradation rates in *E. coli* (59) and in *B. subtilis* (60,61).

In order to engineer an SsrA-tag library for *B. subtilis*, we fused various SsrA-derived peptides to the C-terminus of GFP placed under the control of the P_{veg} promoter and R0 RBS. We first used known functional variants of the SsrA tags LAA (wt), ASV, AAV as a C-terminal wild-type tripeptide and the non-functional *SsrA*-LDD tag as a negative controls (62). In addition, we engineered a SsrA-tag library by randomizing the three last amino-acids of the tag using a reduced 12 amino-acids alphabet (NDT codons: Phe, Leu, Ile, Val, Tyr, His, Asn, Asp, Cys, Arg, Ser, Gly), therefore reducing the library size with no stop codons while conserving an equal representation of each type of amino-acids (9,63). We then integrated the different SsrA-tagged GFP variants into the *B. subtilis* chromosome and measured their expression level by flow cytometry. Since all the SsrA-tagged GFP variants are expressed from the same promoter and RBS, we assumed that the observed differences in fluorescence intensity would be mainly due to differences in protein degradation rates.

It was previously shown in *E. coli* that the degradation rate of SsrA-tagged proteins is higher in stationary than in exponential phase, probably due to an increase in protease concentration (64). We supposed that a similar phenomenon could occur in *B. subtilis*. We thus characterised cell cultures of our SsrA-tag library in exponential and stationary phases. As expected, the strain expressing GFP fused to the non-functional LDD tag (GFP-LDD) had a fluorescence intensity similar to that of strain expressing untagged GFP. Of note, for both untagged GFP and GFP-LDD, the expression level increased about 2-fold in stationary phase, probably because of protein accumulation in the absence of dilution of the cellular content in non-growing cells. In contrast, for most of SsrA-tag variants, an important decrease in GFP abundance was observed, particularly in stationary phase (Figure 4). In comparison to untagged GFP, strains containing LAA, AAV and LVA tags showed about a 2-fold decrease in fluorescence intensity in exponential phase (Figure 4B), and about a 50- to 200-fold decrease in stationary phase (Figure 4C), with cells exhibiting low fluorescence intensity.

Therefore, a higher rate of proteolysis of SsrA-tagged proteins in stationary phase also occurs in *B. subtilis*. Assuming that protease concentration is the same in all the *B. subtilis* strains of our SsrA-tag library, our results show that it is possible to tune the protein degradation rate over at least 2 orders of magnitude depending on the C-terminal SsrA-tag tripeptide sequence.

Given the difference in activity between exponential and stationary phases observed using the degradation tags, we wondered if such variation in expression levels came from an unknown regulation of our promoters. We thus measured expression efficiency for all engineered and characterised constructs (promoter and RBS sets and libraries) in stationary phase. Rank orders of RBSs and promoters in relative expression units were conserved between exponential and stationary phase (Supplementary Figure S7). For promoters as well as for RBS sequences, we observed an average increase in REU between exponential and stationary phase of ∼1.5-fold with a standard deviation of 0.7. Some constructions show a stronger increase in GFP levels (e.g. promoter PY12, PS19, P_{folEA}, RBSs R3, R6, R7). This small global increase could here again be due to the diminution of cell-division related dilution of the cellular content in stationary phase. In conclusion, promoter and RBS libraries can be used to tune gene expression in *B. subtilis* in both exponential and stationary phase, with comparable REU values, and importantly with a conserved rank order.

Assessment of measurements robustness via data comparison between two laboratories

To test the reliability and reproducibility of our measurements processes, we decided to characterise a promoter set in two different laboratories. This comparison method has already proved to be useful in past characterisation work (48). For this reliability experiments, we worked with Casein–Hydrolase media supplemented with glucose (CHG) for two reasons. First, as CHG is richer than M9 minimal media, we supposed it would facilitate lab-to-lab calibration as cells would grow better and faster. Second, we wanted to measure our parts activity in another media.

For this test, we focused on the basic 10 original promoters (Figure 2A, and Supplementary Figure S8). After validating a common experimental protocol, we performed experiments separately using the same strains. We performed data analysis using the same methodology (see materials and methods) and obtained similar results in both laboratories, with comparable REU values and a conserved rank order between promoters. While using a limited number of constructs, these data demonstrate that our library behaviour is relatively reliable when measured by different users in different laboratories.

Interestingly, we compared these results performed in CHG media with our previous data performed in M9 minimal media (Supplementary Figure S9), and found a slight variation at the level of REU perhaps reflecting a different metabolic state of the cells (52,65). This difference could also be due to the fact that measurements were performed using different detection methods (bulk measurement on plate-reader for CHG experiments versus single-cell measurement on a flow-cytometer for M9 experiments). Nevertheless, the rank order of the promoters was well conserved, suggesting that the library can be expected to perform similarly in different growth conditions.

7504 Nucleic Acids Research, 2016, Vol. 44, No. 15

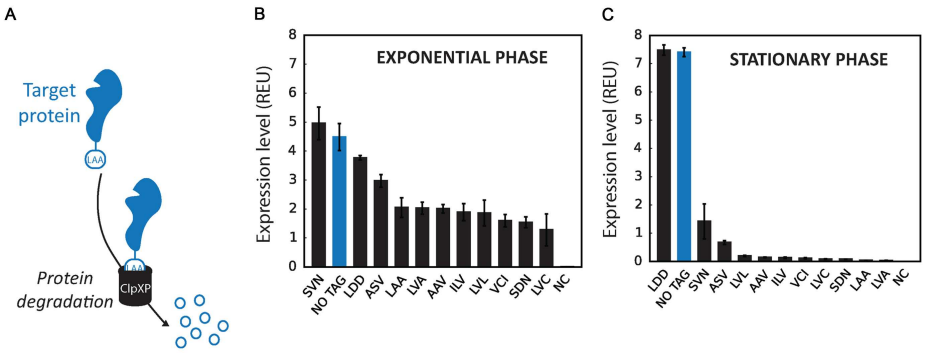


Figure 4. Tuning proteolysis using SsrA degradation tags. (**A**) Principle of SsrA degradation tag: a peptide tag of 14 amino-acids fused the C-terminus of the protein of interest triggers active degradation of the protein by the ClpX protease. Degradation efficiency can be regulated by varying the three C-terminal amino-acids. LAA is the wt sequence and induces strong proteolysis. (**B** and **C**) Library of degradation tags were engineered based on our standardized cassette composed of P_{veg} as promoter, R0 as ribosome binding site and GFP as GOI and following the workflow detailed previously (Figure 2B). The three C-terminal amino-acids of the SsrA tag were randomised (XXX, where X is any of the 12 possible amino-acids of the library). Expression levels of the variants corresponding to three flow-cytometer experiments performed in triplicates in exponential phase (B) or on stationary phase (C) Error bars: ±SD over three independent experiments. Full library measurements data are available in supplementary data files 1 and 5.

Live-cell measurement of the absolute GFP concentration produced by standard parts

Characterisation of our constructs by fluorescence intensity measurements using a flow-cytometer and our reference construct provides convenient calibration and quantification REU. However, for synthetic system design or model prediction, absolute quantification of the number and/or concentration of proteins produced can be desirable (42). For this purpose, we turned to a two-photon (2p) fluorescence fluctuation microscopy method, namely 2p scanning number and brightness (2psN&B) analysis (42). This method was recently adapted for the direct and absolute measurement of fluorescent proteins concentration in individual, live bacterial cells (41). Compared to other microscopy or flow-cytometry methods, the combination of two-photon microscopy and fast raster scan imaging greatly reduces photo-bleaching and background fluorescence, allowing for the precise determination of intracellular concentration of GFP even at very low expression levels (66) (Figure 5A).

For absolute quantification purposes by 2psN&B, the use of monomeric fluorescent proteins is mandatory. If, as often observed, the fluorescent protein reporter tends to aggregate at increasing concentration, its molecular brightness will increase and the molecule numbers will be inaccurately calculated. We thus confirmed that the sfGFP(sp) reporter we used, already described as a monomer (67), remains monomeric in the concentration range of applicability of the 2psN&B method. To do so, we used a transcriptional fusion with the LacI-derived promoter $P_{hyperspank}$ and induced increasing expression of sfGFP(sp) with 0, 5, 10, 20 μM IPTG (Supplementary Figures S1 and S10). We then imaged exponentially growing cells as series of 50 raster-scans and performed N&B analysis (41,42). Although the background-corrected fluorescence intensity values increased over 7 fold, the molecular brightness of the fluorescent particles conserved similar values, averaging at about 0.065 ± 0.04 (counts per molecule per 40 μs dwell time) for induction between 0 to 10 μM of IPTG. This result indicates that the sfGFP(sp) does not self-associates upon increasing intracellular concentration and is therefore a suitable probe for performing 2psN&B experiments. A slightly lower brightness value (0.055) was calculated at 20 μM IPTG, obviously not because of protein aggregation (that would result in an increase of the molecular brightness) but rather because of the high expression level that generates reduced fluctuations of the fluorescent signal, and therefore less accurate determination of the molecular brightness value.

Fluorescence measurement by 2psN&B is a very much time-consuming method and its range of applicability is restricted to low expressed proteins. The 2psN&B method was thus not well suited for the characterisation of our full library of constructs, and we used it with the aim of calibrating expression levels measured by flow cytometry for part expressing low levels of fluorescent proteins. We first selected a set of seven constructs (three from our promoter libraries and four from our RBS libraries) with fluorescence intensities falling into the detection range of 2psN&B. We then measured the fluorescence intensity of single cells containing these different constructs by both 2p fluorescence fluctuations scanning microscopy (Figure 5B) and flow cytometry (Figure 5C). In case of the R1–18 and R2–15 constructs, GFP fluorescence intensity was close to the detection limit of the flow cytometer instrument whereas by 2psN&B it was clearly detected above the auto-fluorescence background level measured in the negative control (NC) strain. Regardless, for all constructs the mean fluorescence intensities measured by the two methods are in very good agreement, providing a linear correlation function relating flow-cytometer fluorescence intensities to the absolute concentration of GFP determined by 2psN&B analysis (Figure 5D). Assuming that this linear relationship remains valid at higher fluorescence intensities, we converted flow cytometry data expressed in relative expression units (REU) in intracellular protein concentration, with one REU (corrected

Nucleic Acids Research, 2016, Vol. 44, No. 15 **7505**

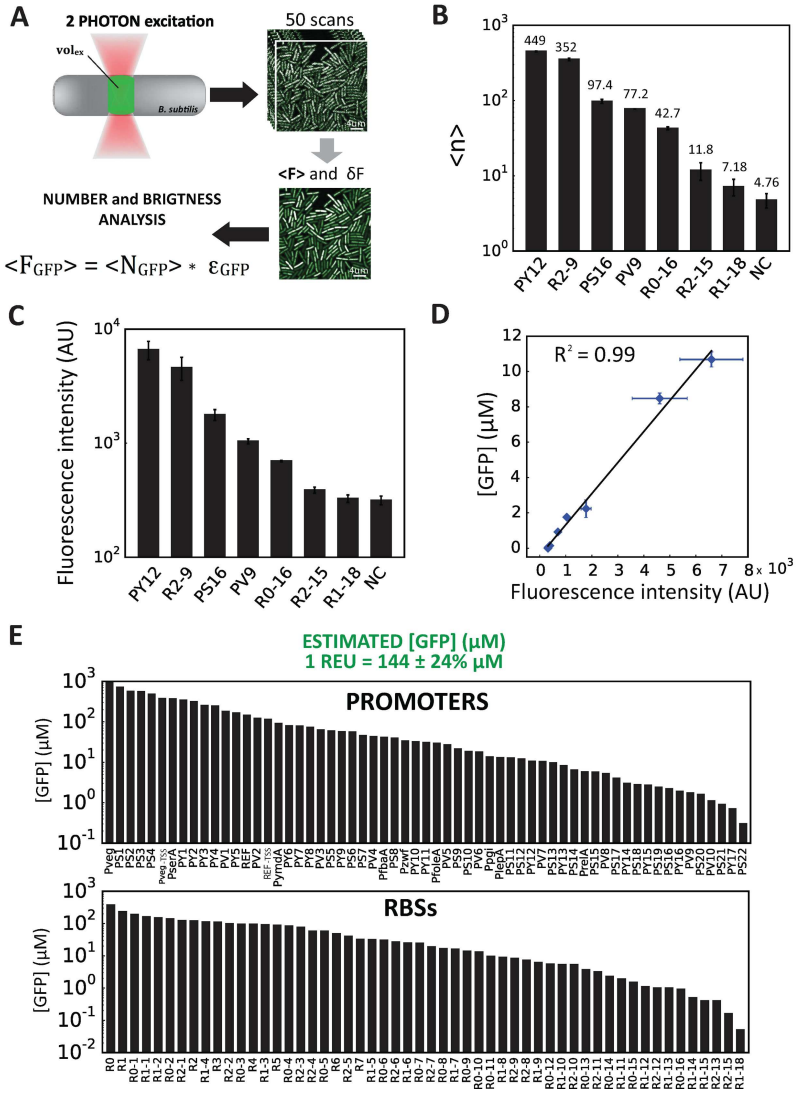


Figure 5. Measurement of part activities at the single molecule level. (**A**) Principle of 2-photon scanning number and brightness (2psN&B) analysis. 2psN&B allows the direct counting of fluorescent molecules diffusing in and out the very small excitation volume generated by 2-photon fluorescence microscopy (see materials and method for details). Bacterial cells expressing GFP and immobilised on an agarose pad are imaged by recording multiple scans (50 scans, 40 μs/pixel). From the mean and variance of the fluorescence signal, the average number of GFP molecules per bacteria and the brightness of the fluorescent protein are calculated. $<F_{GFP>}$: background-corrected mean fluorescence intensity inside cells (A.U.); $<N_{GFP}>$: mean number of GFP molecules; ϵ_{GFP}: average molecular brightness of GFP. (**B**) Number of GFP equivalent molecules per excitation volume (vol_{ex}) produced by different standard parts from the toolbox. Seven constructions spanning the operational range of fluorescence fluctuation measurements were chosen from RBS and promoter libraries. Following the analysis procedure described in materials and methods, the average number of GFP equivalent molecules ($<N>$) detected per vol_{ex} inside the bacterial cells (0.07 fl, about 15% of the cell volume), was calculated for each strain, not corrected for the auto-fluorescence background level measured in the negative control (NC) strain. Data and error bars correspond to the mean and SD of $<N>$ values obtained from three independent experiments. For the R1–18 construct, GFP expression is clearly detected above background, with an average number of GFP molecules ($<N_{GFP}> = <N> - <N_{NC}>$) of 2.4 per vol_{ex}., corresponding to a total of about 16 GFP molecules per cell. (**C**) Fluorescence intensity measurements from the same strains as in (**B**) were performed simultaneously on a flow-cytometer. Note that for the R1–18 construct, GFP expression cannot be detected above background by flow cytometry. (**D**) Linear correlation between the GFP concentration values in μM calculated from 2p sN&B experiments (see materials and methods) and the fluorescence intensity values in arbitrary unit obtained from simultaneous flow-cytometer experiments. Error bars: ±SD over three independent experiments. (**E**) Estimated GFP concentration in μM for all RBS and promoter variants calculated from the following correlation formula obtained from (D): $1\ (REU - REU_{NC}) = 144\ [GFP]\ (\mu M)$. The error on the estimated GFP concentration was estimated to be ∼24%.

7506 Nucleic Acids Research, 2016, Vol. 44, No. 15

for background auto-fluorescence) corresponding approximately to 144 μM of GFP (±24%). As shown in Figure 5E, our complete promoter and RBS library comprising over 135 constructs enable the expression of GFP to be tuned over five orders of magnitude, between concentration ranging from 0.05 to 900 μM, corresponding to an average number of GFP molecules per cell varying from 15 to 270 000.

DISCUSSION

In this work, we provide a well-characterised toolbox to tune gene expression in *B. subtilis* at the level of transcription, translation or proteolysis. We designed a modular and standardised EOU architecture flanked by strong transcriptional terminators to insulate the EOU from the genetic context. Our EOU provides a standard environment for the precise characterisation and comparison of novel biological parts in *B. subtilis.* Using our design, parts can be easily added, deleted, or swapped using restriction enzymes or isothermal Gibson assembly (38), facilitating future reuses and improvements of our libraries. In a future upgrade, our EOU architecture could also be redesigned to support multi-part assembly using Type II restriction enzymes (e.g. Golden gate assembly, (68)).

In order to accelerate the screen for variants exhibiting different properties, we applied a FACS based high-throughput methodology already used in *E. coli* (39). This approach greatly accelerated the design/build/test cycle, and allowed us to rapidly generate three sequence-divergent families of promoters and RBSs. Nucleotide sequence diversity can also possibly be obtained for degradation tags by changing the codon usage of the SsrA peptide. Such sequence variety within our libraries will enable the simultaneous use of multiple components while avoiding recombination problems due to sequence similarities (51,69). The method we describe here can be readily re-applied, if needed, to generate other families of parts with divergent sequences. Parts with similar activities but different sequences could also be used in combination for the expression of multiple genes.

From our various randomization design strategies, we observed that *B. subtilis* is much more stringent than *E. coli* in terms of promoters sequences. While the −35 and −10 boxes of *E. coli* promoters can be directly randomized to obtain a library spanning many orders of magnitude, *B. subtilis* promoters are much subject to have their activity completely abolished by random mutations within these regions. This could be explained by the fact that *B. subtilis* possesses much more different sigma factors, each specific to a growth condition or differentiation stage while *E. coli* has a reduced set of sigma factors (70). On the opposite, mutation within the region between the -35 and -10 boxes are much more tolerated.

Within our promoter libraries, we were able to identify sequences with an improved transcriptional activity and traced back this effect to the reconstitution of a consensus binding sequence for sigma factor SigA (see supplementary data file 1). Of note, we did not observe any effect of our parts on cell growth, even for parts presenting a strong protein expression (Supplementary Figure S11). Interestingly, we also identified variants with an improved activity by generating library of variants in which we randomised the region between the −35 and the −10 boxes. However, the rationale for this increase in transcriptional activities is obscure, but could involve higher-level regulatory effects like DNA looping (71). It would therefore be compelling to expand our approach by combining high-throughput DNA synthesis, FACS and next-generation sequencing (72) to systematically determine the promoter sequence features influencing transcriptional activity in *B. subtilis*.

Regarding context effects, we tested on a small number of sequences the sensitivity of our RBSs activities to two different genes with different sequences, sfGFP and mKate2, and found that two RBSs (4 and 7) had dramatically different activities when used with a different reporter. By incorporating a bicistronic design (57), we were able to partially restore these RBSs function. Our results suggest that context effect can be managed in *B. subtilis* in a similar manner than in *E. coli*. These effects, as well as strategies to mitigate them, need now to be extensively studied. Meanwhile, we provide large enough libraries of parts to quickly circumvent this difficulty.

Interestingly, we observed that the TSS element could also strongly influence the transcriptional activity of the P_{veg} promoter (Supplementary Figure S3). More characterisation is now required to understand the effects of TSS sequences on gene expression, but libraries of TSS sequences could potentially be engineered to provide an additional layer of control of gene expression. From our data, we anticipate that using different TSS sequences could increase the maximal gene expression levels obtained in our libraries. Future work should also be directed to the engineering of well-characterised inducible promoters with various activities and responding to different signals. Finally, the engineering of different integration vectors allowing for simultaneous insertion of multiple gene circuits within the *B. subtilis* chromosome is of utmost utility and should be quickly addressed by the *Bacillus* community.

A significant contribution of our work to the field of biological metrology is the use of fluorescence fluctuation methods to precisely characterise parts activities at the single-molecule level. We were able to identify promoter/RBS combinations producing a concentration as low as 50nM of GFP (∼15 GFP molecules/cell) in exponential phase. By extrapolating our single molecule data over the whole range of our libraries, we estimate that we can tune GFP concentration from nanomolar to millimolar range. Single-molecule measurements are the next frontier in standard parts characterisation, and have recently been explored at the mRNA level for a reference promoter in *E. coli* (69). The systematic development of such approaches promises to improve significantly the precision at which synthetic gene circuits can be tuned, while providing new synthetic tools for researchers investigating the mechanisms regulating gene expression. Engineers will still have to address the challenge of managing noise and stochastic effects in gene expression arising from very low number of molecules.

In conclusion, the libraries of regulatory components presented here are a first step toward a more precise and predictable control of gene expression and dynamics in *B. subtilis.* This toolbox will support many research and engi-

neering applications in the Gram-positive model bacterium, for example for tuning the relative expression levels of various enzymatic members within a synthetic metabolic pathway. All parts and uses demonstrated or disclosed herein have been contributed to the public domain via the BioBrick public agreement (https://biobricks.org/bpa).

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

We thank Emmanuel Margeat, Luca Ciandrini, Joachim Rambeau, Guillaume Cambray, and members of the Bonnet and Jules lab for fruitful discussions. We acknowledge France-BioImaging infrastructure supported by the French National Research Agency (ANR-10-INSB-04-01, ≪ Investments for the future ≫) and the GIS ≪ IBiSA : Infrastructures en Biologie Sante et Agronomie ≫. All sequences and raw data are available in supplementary materials and supplementary data files. All libraries have been deposited and are available at the Bacillus Genetic Stock Center (http://www.bgsc.org/)

FUNDING

French Institut National de la Santé et de la Recherche Médicale (INSERM); Centre National pour la Recherche Scientifique (CNRS); Institut National de la Recherche Agronomique (INRA). S.G. is a recipient of a PhD Fellowship from the French Ministry of Research. INSERM Avenir program, Bettencourt-Schueller Foundation; ERC starting Grant (to J.B.). Funding for open access charge: INSERM (Atip-Avenir program) (to J.B.).
Conflict of interest statement. None declared.

REFERENCES

1. Endy,D. (2005) Foundations for engineering biology. *Nature*, **438**, 449–453.
2. Ro,D., Paradise,E.M., Ouellet,M., Fisher,K.J., Newman,K.L., Ndungu,J.M., Ho,K.A., Eachus,R.A., Ham,T.S., Kirby,J. *et al.* (2006) Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*, **440**, 940–943.
3. Ajikumar,P.K., Xiao,W.-H., Tyo,K.E.J., Wang,Y., Simeon,F., Leonard,E., Mucha,O., Phon,T.H., Pfeifer,B. and Stephanopoulos,G. (2010) Isoprenoid pathway optimization for Taxol precursor overproduction in Escherichia coli. *Science*, **330**, 70–74.
4. Galanie,S., Thodey,K., Trenchard,I.J., Filsinger Interrante,M. and Smolke,C.D. (2015) Complete biosynthesis of opioids in yeast. *Science*, **349**, 1095–1100.
5. Stricker,J., Cookson,S., Bennett,M.R., Mather,W.H., Tsimring,L.S. and Hasty,J. (2008) A fast, robust and tunable synthetic gene oscillator. *Nature*, **456**, 516–519.
6. Toman,Z., Dambly-Chaudiere,C., Tenenbaum,L. and Radman,M. (1985) A system for detection of genetic and epigenetic alterations in Escherichia coli induced by DNA-damaging agents. *J. Mol. Biol.*, **186**, 97–105.
7. Gardner,T.S., Cantor,C.R. and Collins,J.J. (2000) Construction of a genetic toggle switch in Escherichia coli. *Nature*, **403**, 339–342.
8. Ajo-franklin,C.M., Drubin,D.A., Eskin,J.A., Gee,E.P.S., Landgraf,D., Phillips,I. and Silver,P.A. (2007) Rational design of memory in eukaryotic cells service Rational design of memory in eukaryotic cells. *Genes Dev.*, **21**, 2271–2276.
9. Bonnet,J., Subsoontorn,P. and Endy,D. (2012) Rewritable digital data storage in live cells via engineered control of recombination directionality. *Proc. Natl. Acad. Sci. U.S.A.*, **109**, 8884–8889.
10. Bonnet,J., Yin,P., Ortiz,M.E., Subsoontorn,P. and Endy,D. (2013) Amplifying genetic logic gates. *Science*, **340**, 599–603.
11. Brophy,J.A.N. and Voigt,C.A. (2014) Principles of genetic circuit design. *Nat. Methods*, **11**, 508–520.
12. Basu,S., Gerchman,Y., Collins,C.H., Arnold,F.H. and Weiss,R. (2005) A synthetic multicellular system for programmed pattern formation. *Nature*, **434**, 1130–1134.
13. Weber,W. and Fussenegger,M. (2012) Emerging biomedical applications of synthetic biology. *Nat. Rev. Genet.*, **13**, 21–35.
14. Xie,Z., Wroblewska,L., Prochazka,L., Weiss,R. and Benenson,Y. (2011) Multi-input RNAi-based logic circuit for identification of specifc cancer cells. *Science*, **333**, 1307–1312.
15. Roybal,K.T., Rupp,L.J., Morsut,L., Walker,W.J., McNally,K.A., Park,J.S. and Lim,W.A. (2016) Precision tumor recognition by T cells with combinatorial antigen-sensing circuits. *Cell*, **164**, 770–779.
16. Kotula,J.W., Kerns,S.J., Shaket,L.A., Siraj,L., Collins,J.J., Way,J.C. and Silver,P.A. (2014) Programmable bacteria detect and record an environmental signal in the mammalian gut. *Proc. Natl. Acad. Sci. U.S.A.*, **111**, 4838–4843.
17. Danino,T., Prindle,A., Kwong,G.A., Skalak,M., Li,H., Allen,K., Hasty,J. and Bhatia,S.N. (2015) Programmable probiotics for detection of cancer in urine. *Sci. Transl. Med.*, **7**, 289ra84.
18. Courbet,A., Endy,D., Renard,E., Molina,F. and Bonnet,J. (2015) Detection of pathological biomarkers in human clinical samples via amplifying genetic switches and logic gates. *Sci. Transl. Med.*, **7**, 289ra83.
19. Temme,K., Zhao,D. and Voigt,C.A. (2012) Refactoring the nitrogen fixation gene cluster from Klebsiella oxytoca. *Proc. Natl. Acad. Sci. U.S.A.*, **109**, 7085–7090.
20. Ruano-Gallego,D., Alvarez,B. and Fernandez,L.A. (2015) Engineering the Controlled Assembly of Filamentous Injectisomes in E. coli K-12 for Protein Translocation into Mammalian Cells. *ACS Synth. Biol.*, **4**, 1030–1041.
21. Nielsen,A.A., Segall-Shapiro,T.H. and Voigt,C.A. (2013) Advances in genetic circuit design: novel biochemistries, deep part mining, and precision gene expression. *Curr. Opin. Chem. Biol.*, **17**, 878–892.
22. Mutalik,V.K., Guimaraes,J.C., Cambray,G., Mai,Q.A., Christoffersen,M.J., Martin,L., Yu,A., Lam,C., Rodriguez,C., Bennett,G. *et al.* (2013) Quantitative estimation of activity and quality for collections of functional genetic elements. *Nat.Methods*, **10**, 347–353.
23. Lee,M.E., DeLoache,W.C., Cervantes,B. and Dueber,J.E. (2015) A Highly Characterized Yeast Toolkit for Modular, Multipart Assembly. *ACS Synth. Biol.*, **4**, 975–986.
24. Ede,C., Chen,X., Lin,M.Y. and Chen,Y.Y. (2016) Quantitative Analyses of Core Promoters Enable Precise Engineering of Regulated Gene Expression in Mammalian Cells. *ACS Synth Biol.*, doi:10.1021/acssynbio.5b00266.
25. Earl,A.M., Losick,R. and Kolter,R. (2008) Ecology and genomics of Bacillus subtilis. *Trends Microbiol.*, **16**, 269–275.
26. Buescher,J.M., Liebermeister,W., Jules,M., Uhr,M., Muntel,J., Botella,E., Hessling,B., Kleijn,R.J., Le Chat,L., Lecointe,F. *et al.* (2012) Global network reorganization during dynamic adaptations of Bacillus subtilis metabolism. *Science*, **335**, 1099–1103.
27. Nicolas,P., Mader,U., Dervyn,E., Rochat,T., Leduc,A., Pigeonneau,N., Bidnenko,E., Marchadier,E., Hoebeke,M., Aymerich,S. *et al.* (2012) Condition-dependent transcriptome reveals high-level regulatory architecture in Bacillus subtilis. *Science*, **335**, 1103–1106.
28. Losick,R. and Desplan,C. (2008) Stochasticity and cell fate. *Science*, **320**, 65–68.
29. Schultz,D., Wolynes,P.G. and Ben,E. (2009) Deciding fate in adverse times: sporulation and competence in Bacillus subtilis. *Proc. Natl. Acad. Sci. U.S.A.*, **106**, 21027–21034.
30. Süel,G.M., Kulkarni,R.P., Dworkin,J., Garcia-Ojalvo,J. and Elowitz,M.B. (2007) Tunability and noise dependence in differentiation dynamics. *Science*, **315**, 1716–1719.
31. Marbouty,M., Le Gall,A., Cattoni,D.I., Cournac,A., Koh,A., Fiche,J.-B., Mozziconacci,J., Murray,H., Koszul,R. and Nollmann,M. (2015) Condensin- and Replication-Mediated Bacterial

Chromosome Folding and Origin Condensation Revealed by Hi-C and Super-resolution Imaging. *Mol. Cell*, **59**, 588–602.
32. Van Dijl,J.M. and Hecker,M. (2013) Bacillus subtilis: from soil bacterium to super-secreting cell factory. *Microb. Cell Fact.*, **12**, 3.
33. Das,K. and Mukherjee,A.K. (2007) Crude petroleum-oil biodegradation efficiency of Bacillus subtilis and Pseudomonas aeruginosa strains isolated from a petroleum-oil contaminated soil from North-East India. *Bioresour. Technol.*, **98**, 1339–1345.
34. Iwanicki,A., Piątek,I., Stasiłojć,M., Grela,A., Lęga,T., Obuchowski,M. and Hinc,K. (2014) A system of vectors for Bacillus subtilis spore surface display. *Microb. Cell Fact.*, **13**, 30.
35. Radeck,J., Kraft,K., Bartels,J., Cikovic,T., Dürr,F., Emenegger,J., Kelterborn,S., Sauer,C., Fritz,G., Gebhard,S. *et al.* (2013) The Bacillus BioBrick Box: generation and evaluation of essential genetic building blocks for standardized work with Bacillus subtilis. *J. Biol. Eng.*, **7**, 29.
36. Vojcic,L., Despotovic,D., Martinez,R., Maurer,K.H. and Schwaneberg,U. (2012) An efficient transformation method for bacillus subtilis DB104. *Appl. Microbiol. Biotechnol.*, **94**, 487–493.
37. Guérout-Fleury,A.-M., Frandsen,N. and Stragier,P. (1996) Plasmids for ectopic integration in Bacillus subtilis. *Gene*, **180**, 57–61.
38. Gibson,D.G., Young,L., Chuang,R.-Y., Venter,J.C., Hutchison,C.A. and Smith,H.O. (2009) Enzymatic assembly of DNA molecules up to several hundred kilobases. *Nat. Methods*, **6**, 343–345.
39. Kosuri,S., Goodman,D.B., Cambray,G., Mutalik,V.K., Gao,Y., Arkin,A.P., Endy,D. and Church,G.M. (2013) Composability of regulatory sequences controlling transcription and translation in Escherichia coli. *Proc. Natl. Acad. Sci. U.S.A.*, **110**, 14024–14029.
40. Botella,E., Fogg,M., Jules,M., Piersma,S., Doherty,G., Hansen,A., Denham,E.L., Le Chat,L., Veiga,P., Bailey,K. *et al.* (2010) pBaSysBioII: an integrative plasmid generating gfp transcriptional fusions for high-throughput analysis of gene expression in Bacillus subtilis. *Microbiology*, **156**, 1600–1608.
41. Ferguson,M.L., Le Coq,D., Jules,M., Aymerich,S., Declerck,N. and Royer,C.A. (2011) Absolute quantification of gene expression in individual bacterial cells using two-photon fluctuation microscopy. *Anal. Biochem.*, **419**, 250–259.
42. Digman,M.A., Caiolfa,V.R., Zamai,M. and Gratton,E. (2008) The phasor approach to fluorescence lifetime imaging analysis. *Biophys. J.*, **94**, L14–L16.
43. Dosset,P., Rassam,P., Fernandez,L., Espenel,C., Rubinstein,E., Margeat,E. and Milhiet,P.E. (2016) Automatic detection of diffusion modes within biological membranes using back-propagation neural network. *BMC Bioinformatics*, **17**, 197.
44. Cho,B.-K., Zengler,K., Qiu,Y., Park,Y.S., Knight,E.M., Barrett,C.L., Gao,Y. and Palsson,B.Ø. (2009) Elucidation of the transcription unit architecture of the Escherichia coli K-12 MG1655 genome. *Nat. Biotechnol.*, **27**, 1043–1049.
45. Krasny,L., Tiserova,H., Jonak,J., Rejman,D. and Sanderova,H. (2008) The identity of the transcription +1 position is crucial for changes in gene expression in response to amino acid starvation in Bacillus subtilis. *Mol. Microbiol.*, **69**, 42–54.
46. Sojka,L., Kouba,T., Barvik,I., Sanderova,H., Maderova,Z., Jonak,J. and Krasny,L. (2011) Rapid changes in gene expression: DNA determinants of promoter regulation by the concentration of the transcription initiating NTP in Bacillus subtilis. *Nucleic Acids Res.*, **39**, 4598–4611.
47. Overkamp,W., Beilharz,K., Detert Oude Weme,R., Solopova,A., Karsens,H., Kovacs,A.T., Kok,J., Kuipers,O.P. and Veening,J.-W. (2013) Benchmarking various green fluorescent protein variants in Bacillus subtilis, Streptococcus pneumoniae, and Lactococcus lactis for live cell imaging. *Appl. Environ. Microbiol.*, **79**, 6481–6490.
48. Kelly,J.R., Rubin,A.J., Davis,J.H., Ajo-Franklin,C.M., Cumbers,J., Czar,M.J., de Mora,K., Glieberman,A.L., Monie,D.D. and Endy,D. (2009) Measuring the activity of BioBrick promoters using an in vivo reference standard. *J. Biol. Eng.*, **3**, 4.
49. Lam,K.H.E., Chow,K.C. and Wong,W.K.R. (1998) Construction of an efficient Bacillus subtilis system for extracellular production of heterologous proteins. *J. Biotechnol.*, **63**, 167–177.
50. Keasling,J.D. (2012) Synthetic biology and the development of tools for metabolic engineering. *Metab. Eng.*, **14**, 189–195.
51. Canton,B., Labno,A. and Endy,D. (2008) Refinement and standardization of synthetic biological parts and devices. **26**, 787–793.
52. Borkowski,O., Goelzer,A., Schaffer,M., Calabre,M., Mäder,U., Aymerich,S., Jules,M. and Fromion,V. (2016) Translation elicits a growth rate-dependent, genome-wide, differential protein production in *Bacillus subtilis*. *Mol. Syst. Biol.*, **12**, 870.
53. Nakagawa,S., Niimura,Y., Miura,K. and Gojobori,T. (2010) Dynamic evolution of translation initiation mechanisms in prokaryotes. *Proc. Natl. Acad. Sci. U.S.A.*, **107**, 6382–6387.
54. Doi,R.H. (1984) Genetic Engineering in Bacillus subtilis. *Biotechnol. Genet. Eng. Rev.*, **2**, 121–155.
55. Vellanoweth,R.L. and Rabinowitz,J.C. (1992) The influence of ribosome-binding-site elements on translational efficiency in Baciiius subtiis and Escherichia coii in vivo. *Mol. Microbiol.*, **6**, 1105–1114.
56. Shcherbo,D., Murphy,C.S., Ermakova,G. V, Solovieva,E. a, Chepurnykh,T. V, Shcheglov,A.S., Verkhusha,V. V, Pletnev,V.Z., Hazelwood,K.L., Roche,P.M. *et al.* (2009) Far-red fluorescent tags for protein imaging in living tissues. *Biochem. J.*, **418**, 567–574.
57. Mutalik,V.K., Guimaraes,J.C., Cambray,G., Lam,C., Christoffersen,M.J., Mai,Q.-A., Tran,A.B., Paull,M., Keasling,J.D., Arkin,A.P. *et al.* (2013) Precise and reliable gene expression via standard transcription and translation initiation elements. *Nat. Methods*, **10**, 354–360.
58. Elowitz,M.B. and Leibler,S. (2000) A synthetic oscillatory network of transcriptional regulators. *Nature*, **403**, 335–338.
59. Andersen,J.B., Sternberg,C., Poulsen,L.K., Bjørn,S.P., Givskov,M. and Molin,S. (1998) New unstable variants of green fluorescent protein for studies of transient gene expression in bacteria. *Appl. Environ. Microbiol.*, **64**, 2240–2246.
60. Wiegert,T. and Schumann,W. (2001) SsrA-mediated tagging in Bacillus subtilis SsrA-mediated tagging in Bacillus subtilis. *J. Bacteriol.*, **183**, 3885–3889.
61. Griffith,K.L. and Grossman,A.D. (2008) Inducible protein degradation in Bacillus subtilis using heterologous peptide tags and adaptor proteins to target substrates to the protease ClpXP. *Mol. Microbiol.*, **70**, 1012–1025.
62. McGinness,K.E., Baker,T.A. and Sauer,R.T. (2006) Engineering controllable protein degradation. *Mol. Cell*, **22**, 701–707.
63. Reetz,M.T. and Wu,S. (2008) Greatly reduced amino acid alphabets in directed evolution: making the right choice for saturation mutagenesis at homologous enzyme positions. *Chem. Commun. (Camb).*, doi:10.1039/b813388c.
64. Farrell,C.M., Grossman,A.D. and Sauer,R.T. (2005) Cytoplasmic degradation of ssrA-tagged proteins. *Mol. Microbiol.*, **57**, 1750–1761.
65. Klumpp,S. and Hwa,T. (2008) Growth-rate-dependent partitioning of RNA polymerases in bacteria. *Proc. Natl. Acad. Sci. U.S.A.*, **105**, 20245–20250.
66. Ferguson,M.L., Le Coq,D., Jules,M., Aymerich,S., Radulescu,O., Declerck,N. and Royer,C.A. (2012) Reconciling molecular regulatory mechanisms with noise patterns of bacterial metabolic promoters in induced and repressed states. *Proc. Natl. Acad. Sci. U.S.A.*, **109**, 155–160.
67. Pédelacq,J.-D., Cabantous,S., Tran,T., Terwilliger,T.C. and Waldo,G.S. (2006) Engineering and characterization of a superfolder green fluorescent protein. *Nat. Biotechnol.*, **24**, 79–88.
68. Engler,C., Kandzia,R. and Marillonnet,S. (2008) A one pot, one step, precision cloning method with high throughput capability. *PLoS One*, **3**, e3647.
69. Nielsen,A.A.K., Der,B.S., Shin,J., Vaidyanathan,P., Paralanov,V., Strychalski,E.A., Ross,D., Densmore,D. and Voigt,C.A. (2016) Genetic circuit design automation. *Science*, **352**, aac7341.
70. Artsimovitch,I., Svetlov,V., Anthony,L. and Burgess,R.R. (2000) RNA polymerases from Bacillus subtilis and Escherichia coli differ in recognition of regulatory signals in vitro. **182**, 6027–6035.
71. Cournac,A. and Plumbridge,J. (2013) DNA looping in prokaryotes: experimental and theoretical approaches. *J. Bacteriol.*, **195**, 1109–1119.
72. Sharon,E., Kalma,Y., Sharp,A., Raveh-Sadka,T., Levo,M., Zeevi,D., Keren,L., Yakhini,Z., Weinberger,A. and Segal,E. (2012) Inferring gene regulatory logic from high-throughput measurements of thousands of systematically designed promoters. *Nat. Biotechnol.*, **30**, 521–530.

Supplementary materials for:

A part toolbox to tune genetic expression in *Bacillus subtilis.*

Authors: Sarah Guiziou[1], Vincent Sauveplane[2], Hung-Ju Chang[1], Caroline Clerté[1], Nathalie Declerck[1], Matthieu Jules[2], and Jerome Bonnet[1*]

Affiliations:

[1]Centre de Biochimie Structurale, INSERM U1054, CNRS UMR5048, University of Montpellier, France.

[2] Micalis Institute, INRA, AgroParisTech, Université Paris-Saclay, 78350 Jouy-en-Josas, France

*To whom correspondence should be addressed.

Email: jerome.bonnet@inserm.fr

These supplementary materials contain:

-Supplementary methods.

-Supplementary Figures S1 to S11.

-DNA sequences of the EOU, the mkate2 protein, the BCDs, the spacers, the primers and G-Blocks used in this study.

These supplementary materials are accompanied by supplementary data files containing parts sequences and raw measurements data:

-Supplementary data file 1 contains sequences for all promoters, RBSs, and Ssra tags, as well as mean values and standard deviation of their relative activities, expressed in arbitrary units and in REUs and estimated GFP concentrations.

- Supplementary data file 2 contains raw data values from all cytometry experiments for parents promoters and RBSs, in exponential and stationary phase, expressed in arbitrary units and in REUs.

- Supplementary data file 3 contains raw data values from all cytometry experiments for promoter libraries, in exponential and stationary phase, expressed in arbitrary units and in REUs.

- Supplementary data file 4 contains raw data values from all cytometry experiments for RBS libraries, in exponential and stationary phase, expressed in arbitrary units and in REUs.

- Supplementary data file 5 contains raw data values from all cytometry experiments for SsrA tags libraries, in exponential and stationary phase, expressed in arbitrary units and in REUs.

- Supplementary data file 6 contains raw data values for cytometry experiments comparing context effect using GFP or mkate2 as a reporter, with and without BCD. Data are expressed in arbitrary units and in REUs.

- Supplementary data file 7 contains raw data values for plate-reader experiments comparing promoter activities between Montpellier and Jouy-en–Josas.

Supplementary methods

Bacillus subtilis integration protocol

Bacillus strain 168 was streaked on an LB agar plate. 5 mL of Medium A (0.2% ammonium sulfate, 1.4% dipotassium hydrogen phosphate, 0.6% potassium dihydrogen phosphate, 0.07% sodium citrate, 0.5% glucose, 0.02% magnesium sulfate heptahydrate, 0.2% yeast extract, and 0.025% casamino acids) was inoculated at 0.2-0.25 OD_{600nm} from fresh streaked plate and incubated at 37°C. After cessation of log growth (according to semi-log plot of OD_{600nm} over time) and 90 supplementary minutes, 0.5 mL of the culture was transferred into 4.5 mL of pre-warmed Medium B (0.2% ammonium sulfate, 1.4% dipotassium hydrogen phosphate, 0.6% potassium dihydrogen phosphate, 0.07% sodium citrate, 0.5% glucose, 0.08% magnesium sulfate heptahydrate, 0.1% yeast extract, 0.01% casamino acids, and 0.05% calcium chloride). After 90 min at 37°C, the culture should be highly competent. Competent cells were stored at -80°C with 1/10 v/v of glycerol, aliquoted at 0.5 mL in sterile tube and frozen with liquid nitrogen. For integration, tubes of 0.5 mL of competent cells were thaw at 37°C, 500 ng of DNA was added to the competent cells and incubated at 37°C for 30 minutes. 200 μL of LB was added to cells and cultures were incubated at 37°C for at least 30 minutes. Cells were plated on selective agar plated after spinning at 4 000 rpm during 2 minutes and removed of 600 μL of supernatant.

Plasmid constructions

For plasmid construction and amplification, *E. coli* NEB10β strain (NEB) was used. Transformation of plasmids was accomplished by electroporation. *E. coli* were made electrocompetent following the instruction manual of Bio-rad MicroPulser. Electroportations were realized adding 1 µL of DNA to 40 µL of competent cells and using Bio-Rad GenePulser and Ec1 program with 0.1 cm cuvette. SOC media was added after electric shock and cells were grown one hour at 37°C before plated on selective media. Antibiotics used were Ampicillin (100 µg/mL), Spectinomycin (100 µg/mL) and Erythromycin (0.5 µg/mL) (Sigma).

Design and construction of EOU (expression operating unit)
Two basic synthetic EOUs for gene expression were designed, one with sfGFP(sp) as reporter (47) and the other one with mKate2 (56) optimized for *B. subtilis* (with DNA2.0 algorithm). Both EOUs were composed of identical 40 base pair spacers design for Gibson Assembly (designed using the R2odna software, http://www.r2odna.com) to allow simple construction and switch of parts, of restriction enzyme sites for library constructions, of bidirectional and double terminators to insulate from genetic context (Bba_B0014 and Bba_B0015) and of parts for gene expression in *B. subtilis*, P_{veg} promoter mutated to add

AgeI restriction site, R0 RBS and mKate2(Bs) (simply named RFP from here) or sfGFP(sp) (simply named GFP from here). Cassettes named P_{veg}.RFP and P_{veg}.GFP were synthetised by DNA2.0 (Menlo Park, CA, USA).

P_{veg}.RFP and P_{veg}.GFP were cloned inside pDG1730 vector between position 636 to 676 using Gibson Assembly. pDG1730 and DNA2.0 fragments were PCR amplified with respectively P31/P32 and P33/P34 and ligations were performed using iso-thermal Gibson Assembly. After transformation using electroporation (Ampicillin selection), colony PCR verification (One Taq master mix and P16/P34 primers) and plasmid extraction, constructs were sequence for validation and named SG11 (RFP) and SG13 (GFP).

To construct SG29: P_{veg}.GFP with TSS and SG30: P_{REF}.GFP with TSS, part TSS was added to respectively SG13 and SG22 by Gibson Assembly with PCR amplification of vector by P162/P39 and of insert by P137/P34. SG22 was previously obtained by randomization of SG13 using P64 primer (see promoter libraries section).

Set of 10 promoters
For construction of the set of 10 promoters, 10 Gblocks were designed to replace P_{veg} promoters by the one of interest based on SG29. SG29 was PCR amplified with P109/P36 and ligations with Gblocks were performed using iso-thermal Gibson Assembly.

Set of 8 RBSs, with GFP and RFP as reporter
For construction of the set of 8 RBSs with GFP, primers composed of new RBSs and around 20bp homology sequence with GFP were designed. SG29 was PCR amplified as vector with P146/P39 and SG29 was PCR amplified with primers specific for each RBS (R1 to R7: P139 to P145 and R8: P225), ligations were performed using iso-thermal Gibson Assembly.

For construction of the set of 8 RBSs with RFP, SG29 was PCR amplified as vector with P146/P72 and SG11 was PCR amplified as insert with P34 and primers specific for each RBS (R0: P238, R1 to R7: P206 to P212). Ligations were performed using iso-thermal Gibson Assembly and constructions were sequenced using P33/P34 primers.

Set of 4sSsrA tags
For construction of a set of 4 classic SsrA tags, 4 Gblocks were designed to add SsrA degradation tags to SG13. SG13 was PCR amplified by P124/P39 and ligations with Gblocks were performed using iso-thermal Gibson Assembly.

Promoter, RBS and SsrA tag library construction

The various promoters, RBS and degradation tags libraries were generated by performing a PCR on the GFP gene using primers contain the regulatory region of interest

degenerated at strategic positions.

Promoter libraries

For the initial P_{veg} libraries (Figure S2A), SG13 was randomized by PCR amplification using P34 and the degenerated primers: P62 (full randomization), P63 (randomization of -35 box) or P64 (randomization of -10 box). For final P_{veg} libraries, SG29 was randomized by PCR amplification using P34 and the degenerated primers P213. Vectors and amplified fragments were digested by *Age*I and *Sph*I. For P_{serA} and P_{ymdA} libraries, SG36 (cassette with P_{serA} as promoter) and SG37 (P_{ymdA} as promoter) were randomized by PCR amplification using P34 and respectively P214 and P215 for design 1 (randomisation of -10 box), P341 and P342 for design 2 (randomisation of 6 nucleotides before -10 box). Vectors and amplified fragments were digested by *Bam*HI and *Sph*I.

RBS libraries

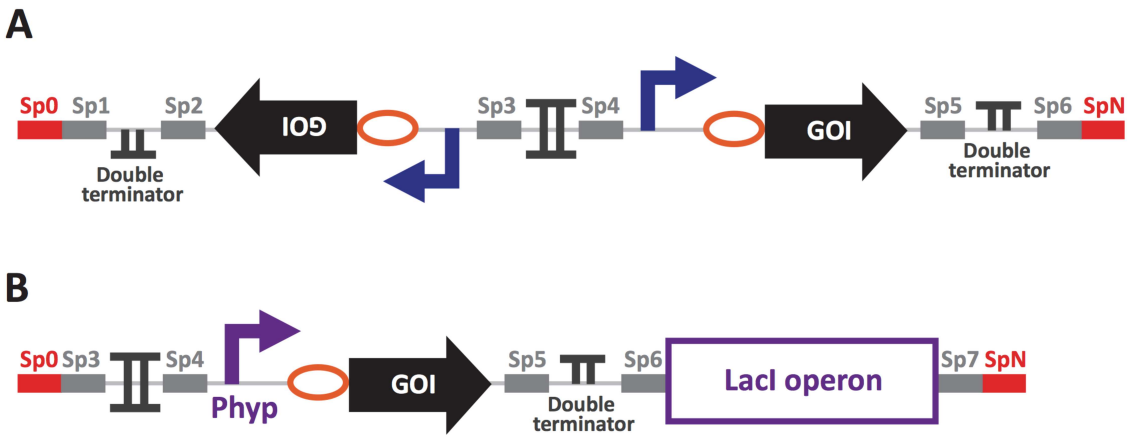
For RBS libraries, SG29 (cassette with R0 as RBS), SG45 (R1 as RBS) and SG47 (R2 as RBS) were randomized by PCR amplification using P34 and the degenerated primers respectively: P220, P221 and P222. Vectors and amplified fragments were restricted by NheI and SphI.

SsrA tag libraries

For SsrA tag libraries, SG13 was randomized by PCR amplification using P33 and P51 degenerated primer. Vectors and amplified fragments were restricted by *Nhe*I and *Sph*I.

Supplementary figures

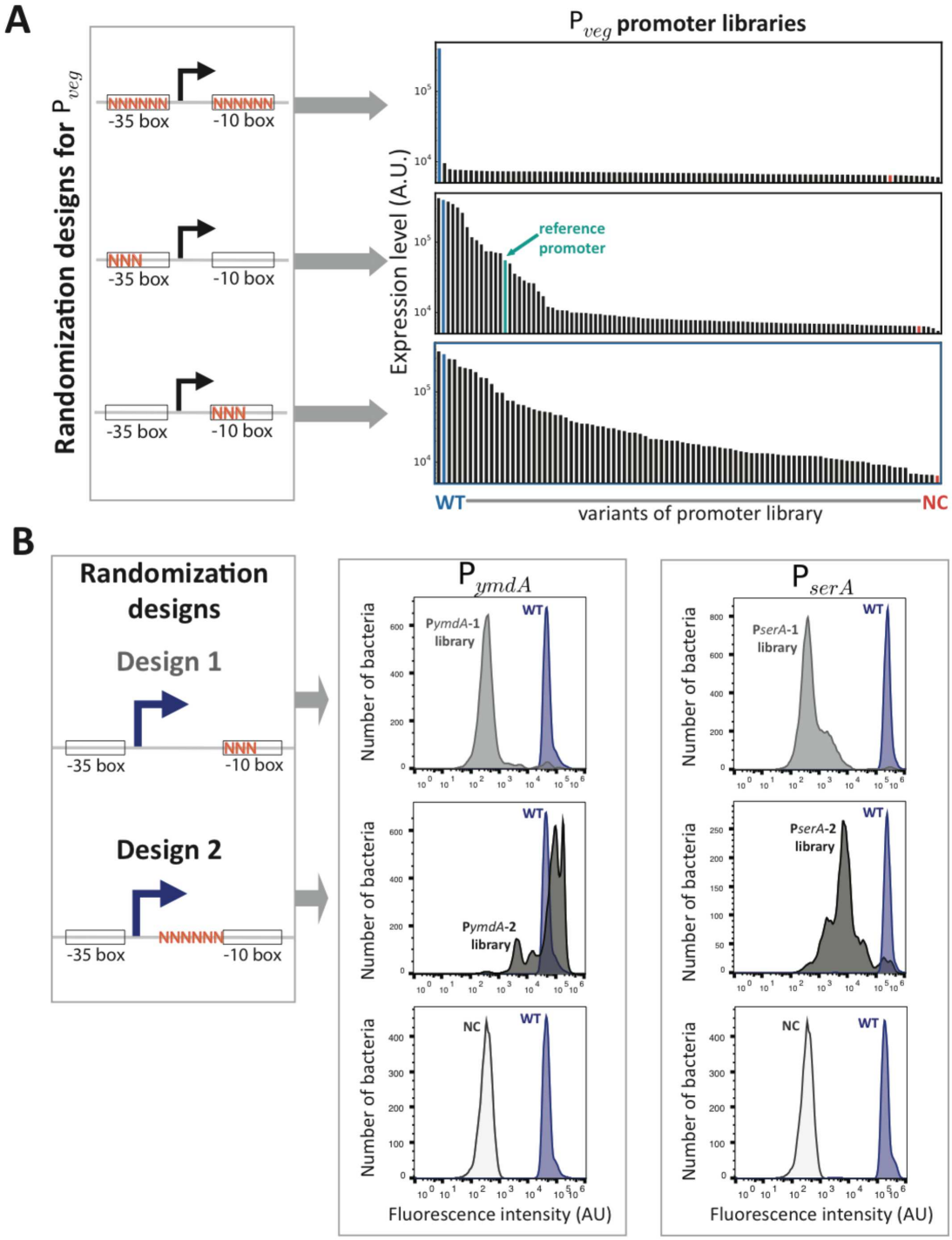
Figure S1: Expression operating unit design for expression of two genes and for inducible expression of gene.



(A) Architecture of our standardized and modular double expression operating unit (dEOU) based on the same design than EOU. The dEOU is composed of two sets of standard regulatory elements (promoter, RBS, GOI) positioned in opposite direction two maximize insulation between the two cassettes. Spacers (SpX) of 40 bp designed to facilitate for one-step isothermal assembly enable simple construction and switching of parts.

(B) Architecture of expression operating unit for inducible expression with example of $P_{hyperspank}$ promoter expressed under regulation of LacI repressor with induction by IPTG. Repressor operon is placed in 3' of the EOU, between spacer Sp6 and Sp7.

Figure S2: Engineering promoter libraries: different parent sequences and randomisation designs result in libraries with various distribution of expression level.

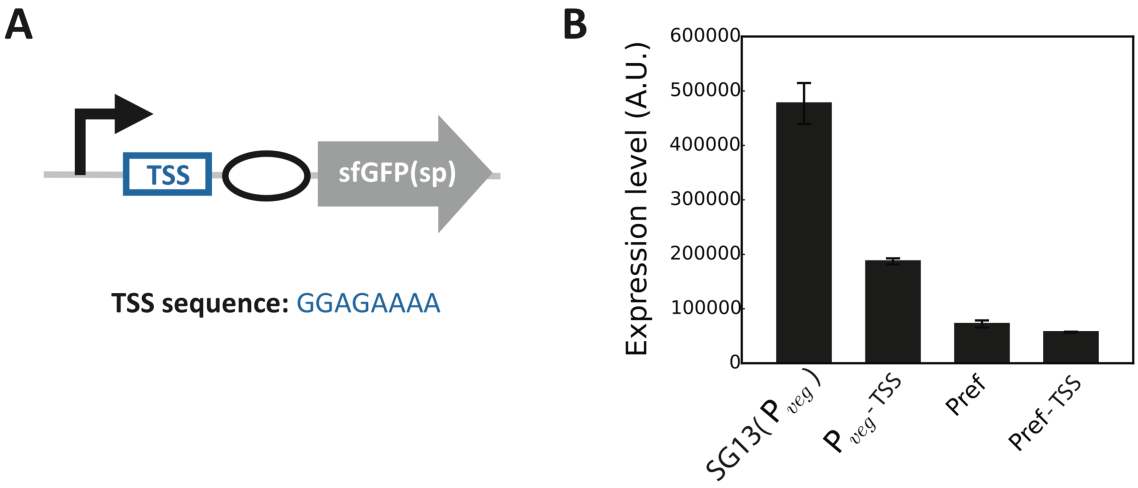


(A) Based on P_{veg} sequences (without TSS), we tested 3 promoter libraries designs (left panel); randomisation of -35 box and -10 box, randomisation of 3 nucleotides within the -10 box and randomisation of 3 nucleotides within the -35 box. Distribution of expression levels for each library (right panel) corresponds to the measurement of fluorescence

intensity over OD on a plate reader (see methods for details). The selected reference promoter obtained from the -10 box randomisation library is highlighted in pale blue.

(B) Two other parent promoters: P_{serA} and P_{ymdA} were used for engineering promoter libraries. We tested 2 designs (left panel): randomisation of 3 nucleotides within the -10 box and randomisation of 6 nucleotides between the -35 and -10 box. Right panel, distribution of expression level for the 4 libraries (2 promoters parent sequences and 2 randomisation designs for each of them). Histograms are obtained by measurement of the fluorescence intensity (in arbitrary units) of the pool of variants by flow-cytometry after integration in *B. subtilis*. For details, see method.

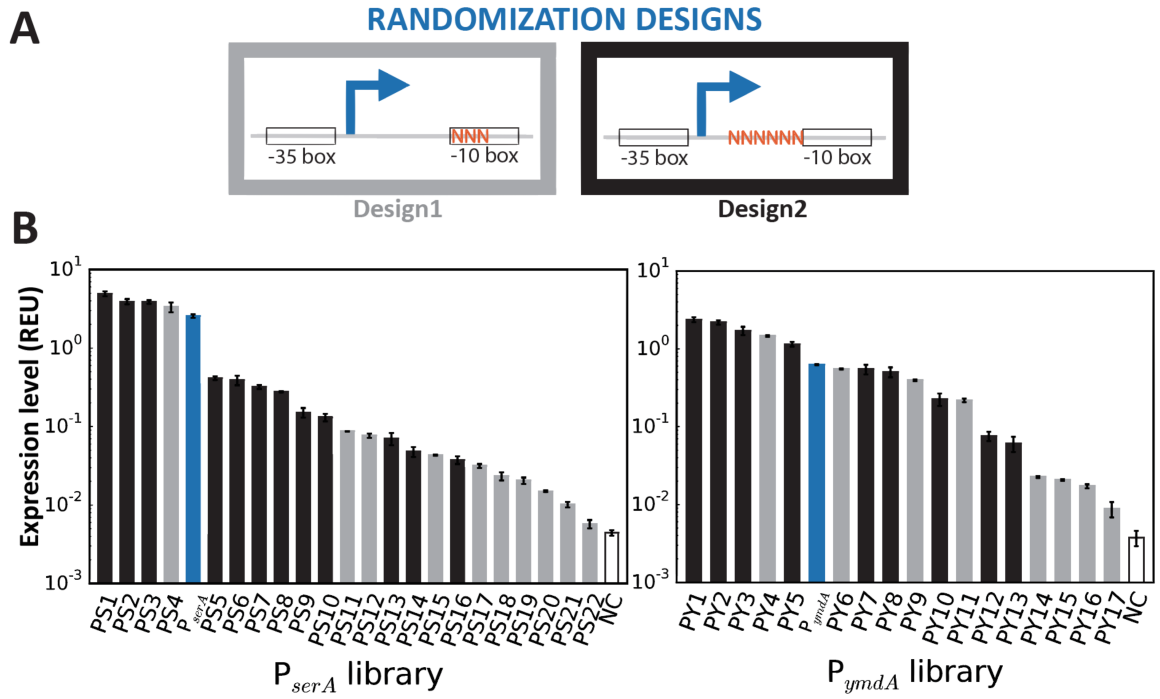
Figure S3: Effect of TSS part on promoter efficiency.



(A) A standard TSS sequence was placed between promoters and ribosome binding sites. We chose the 8 nucleotides after the promoter of *fbaA* gene of *B. subtilis*.

(B) Expression level of 2 promoters with and without the standard TSS part; SG13 (corresponding to the P_{veg} promoter without the TSS part), P_{veg} (TSS), the reference promoter P_{REF} (a variant of P_{veg}), and P_{REF} (TSS). Constructs were cloned on our modular cassette (R0 was used as a RBS). Expression levels are in arbitrary unit (A.U.) and correspond to the fluorescence intensity measured flow-cytometer obtained over 3 independent experiments performed on exponential phase (3 replicates per experiments). Error-bars correspond to standard deviation between the 3 experiments.

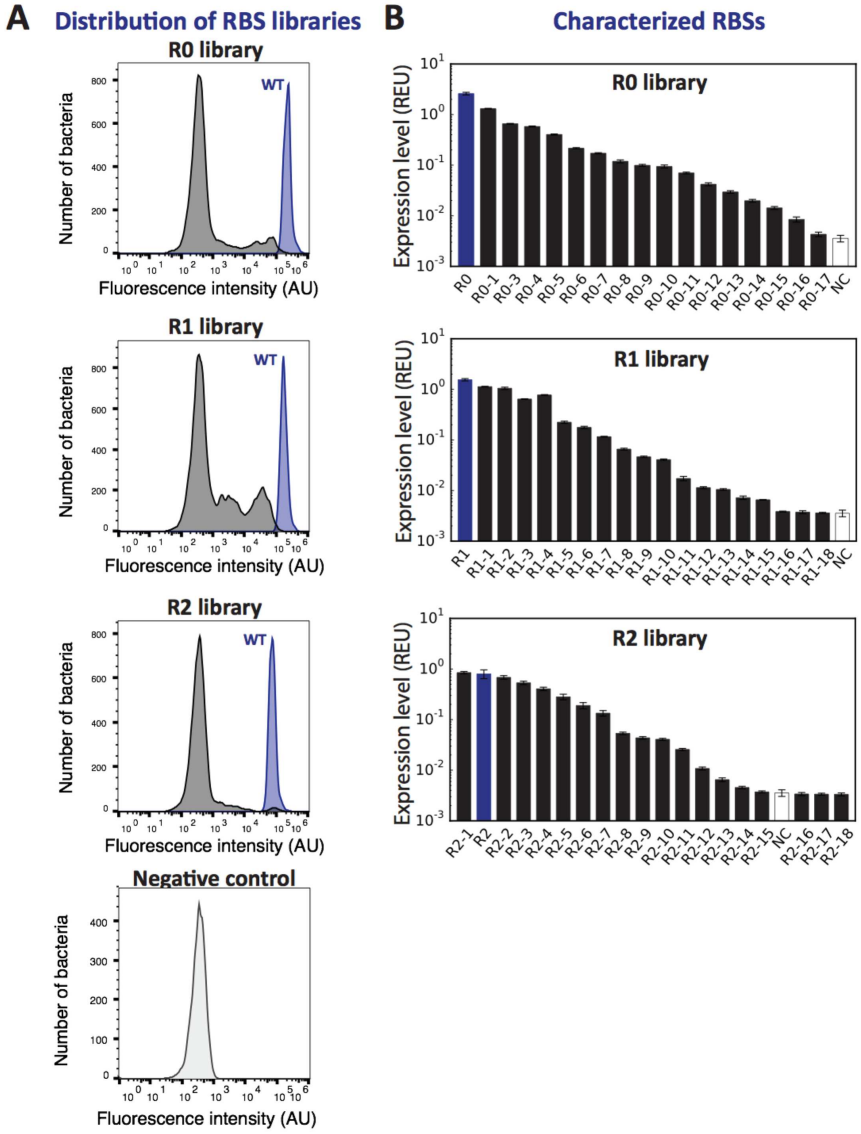
Figure S4: Fully characterised P_{serA} and P_{ymdA} variants and comparison of the two different randomisation designs.



(A) Two different randomisation designs, design 1: randomisation of 3 nucleotides within the -10 box (grey) and design 2: 6 nucleotides between the -35 box and the -10 box (black).

(B) (C) Characterisation of over 10 variants per design for P_{serA} libraries **(B)** and P_{ymdA} libraries **(C)**. Expression levels were measured by flow-cytometry and expressed in REU. Data correspond to the mean of 3 independent experiments performed in triplicates and error-bars correspond to standard deviation over these 3 experiments. Grey bars design variants engineered following the design 1 and black bars design variants engineered following the design 2.

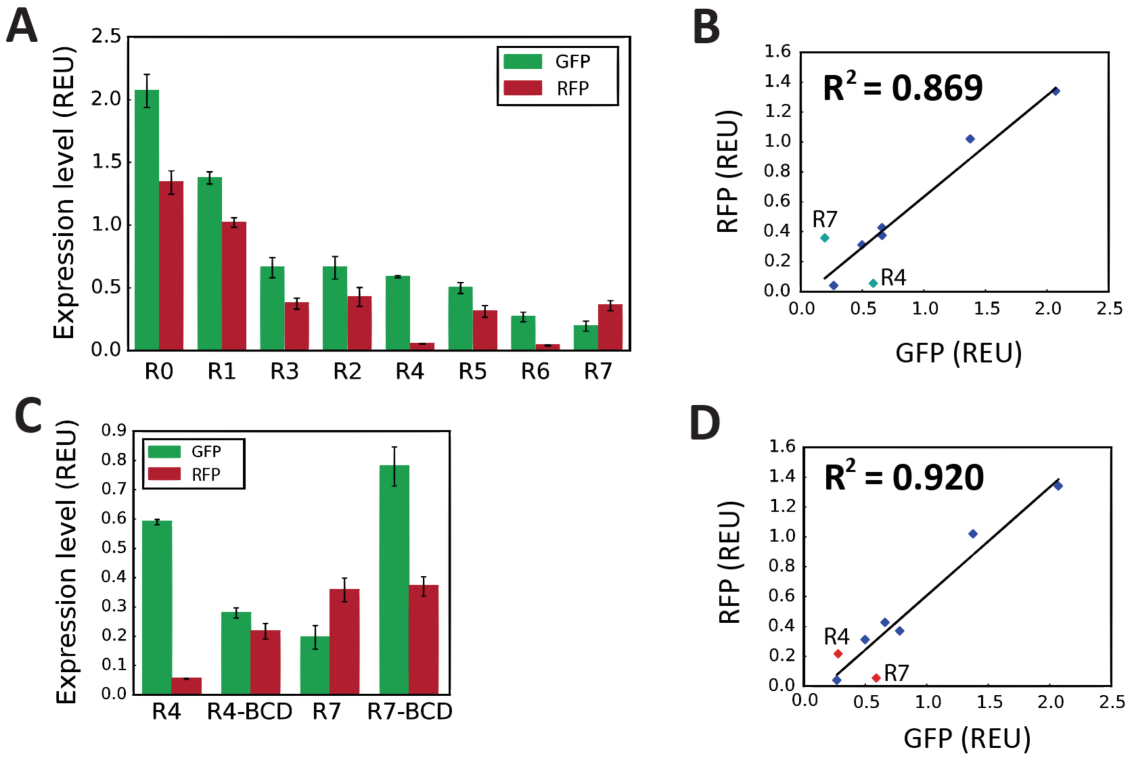
Figure S5: Distribution of expression levels and full set of characterised variants for the 3 RBS libraries.



(A) Distribution of expression level for the 3 RBS libraries (R0, R1, R2) and the negative control stain (without GFP) as a background fluorescence control. Histograms are obtained by flow-cytometer measurement of fluorescence intensity (in arbitrary unit) of the pool of variants after integration in *B. subtilis*. Blue histogram corresponds to the wild-type RBS sequence and the black histogram to the full library.

(B) Full set of characterised variants for each library. Expression levels were measured by flow-cytometry and expressed in REU. Data correspond to the mean of 3 independent experiments performed in triplicates and error-bars correspond to standard deviation over these 3 experiments.

Figure S6: Effect of reporter sequences on translation efficiency and insulation of this effect using bicistrons.



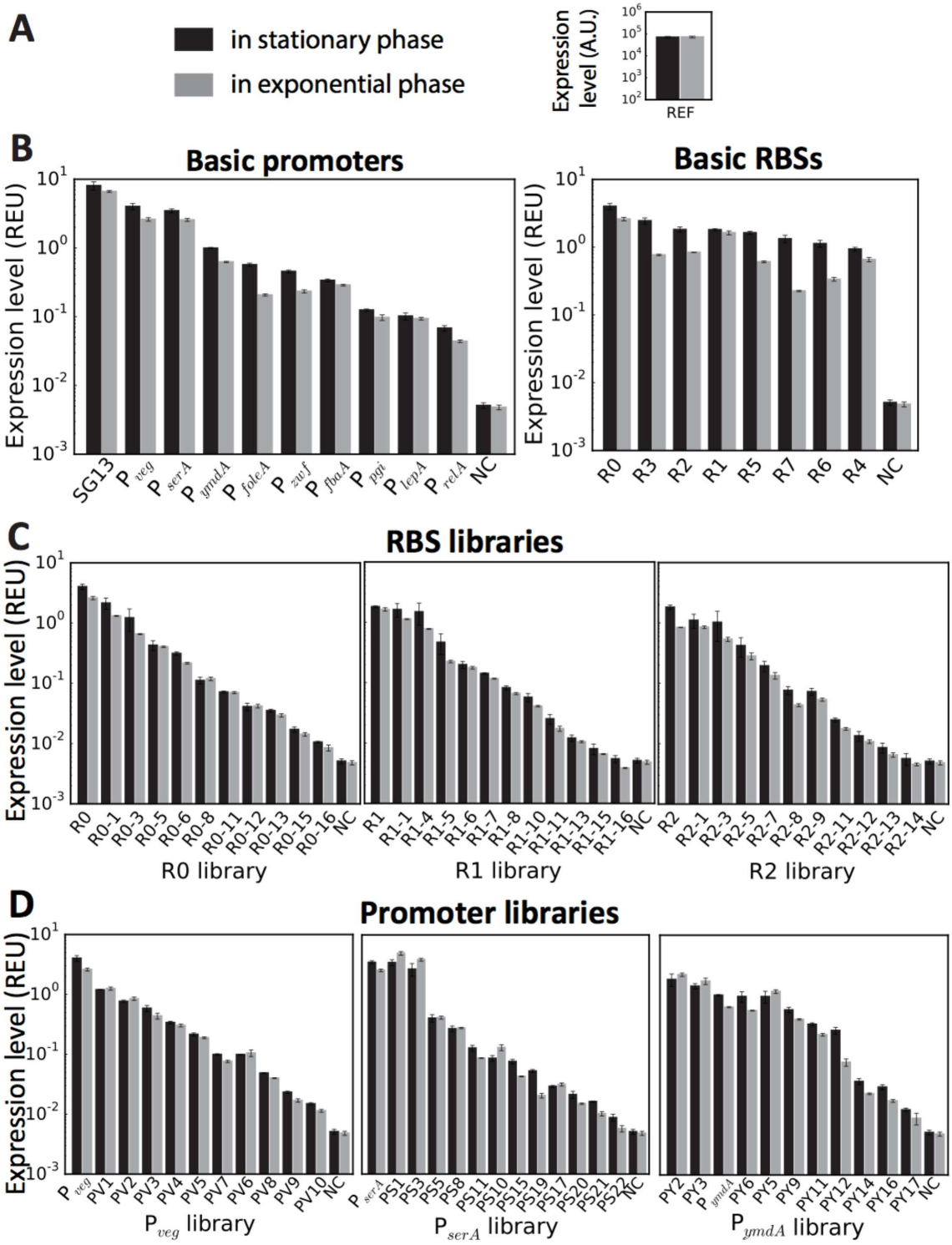
(A) Characterisation of 8 RBSs with two different reporters: sfGFP(sp) (named GFP) and mKate2(Bs) (named RFP). Expression levels were measured by flow-cytometry and expressed in REU. Data correspond to the mean of 3 independent experiments performed in triplicates and error-bars correspond to standard deviation over these 3 experiments. The reference constructs used were P_{REF} with corresponding reporter.

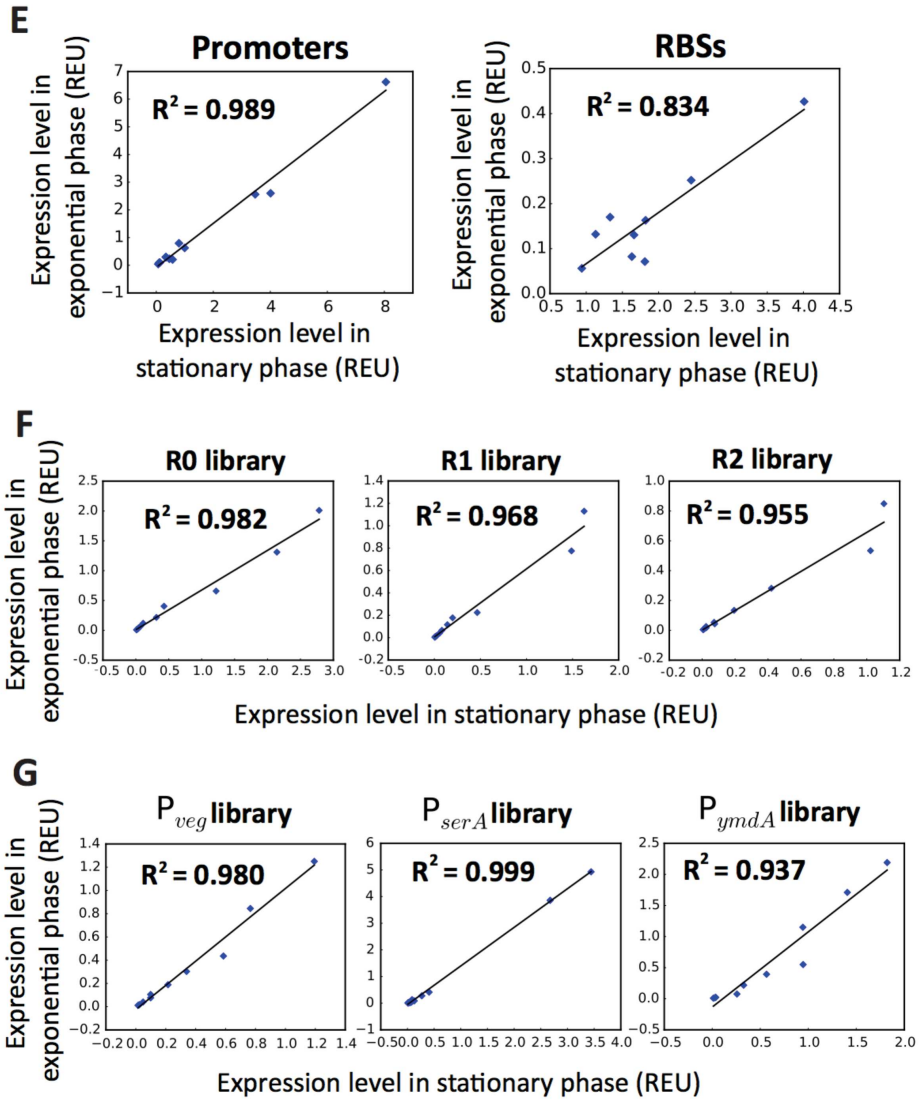
(B) Expression level of RBSs with RFP over expression level of RBSs with GFP in relative expression unit. A linear correlation was found with a coefficient of determination of 0.898. The two ribosome binding sites with the worse correlation are R4 and R7 (green dots).

(C) For R4 and R7, a bicistronic design was used to decouple translation initiation from putative context effects arising from interactios between the RBS and the reporter coding sequence. Expression levels were measured by flow-cytometry and expressed in REU. Data correspond to the mean of 3 independent experiments performed in triplicates and error-bars correspond to standard deviation over these 3 experiments.

(D) Expression level of RBSs coupled with RFP over expression level of RBSs coupled with GFP in REU with R4-BCD and R7-BCD (red dots) instead of mono-cistron constructs. A linear correlation was found with a coefficient of determination of 0.920. Full measurements data are available in supplementary data file 6.

Figure S7: Comparison of promoter and RBS strengths between stationary and exponential phase.

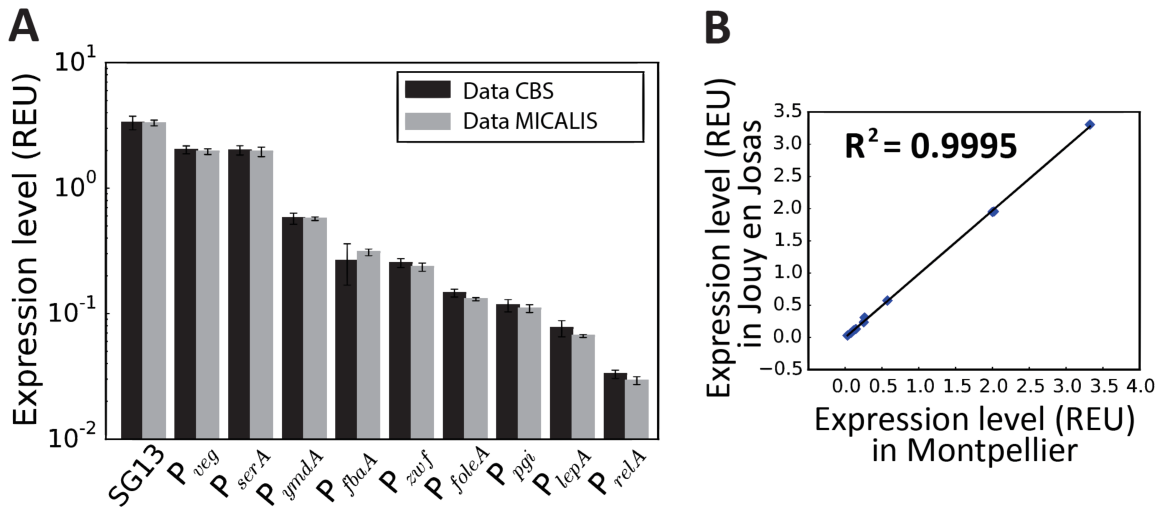




Characterisation of promoter and RBS libraries was performed on exponential and stationary phases. Expression levels were measured by flow-cytometry and expressed in REU. Data correspond to the mean of 3 independent experiments performed in triplicates and error-bars correspond to standard deviation over these 3 experiments. Grey bars correspond to exponential phase and black bars to stationary phase. **(A)** Expression level of the reference construct in absolute unit in exponential and stationary phase. **(B) (C) (D):** Expression levels of basic promoters and RBSs in exponential and stationary phase expressed in REU **(B)**, RBS libraries **(C)** and promoter libraries **(D)**.

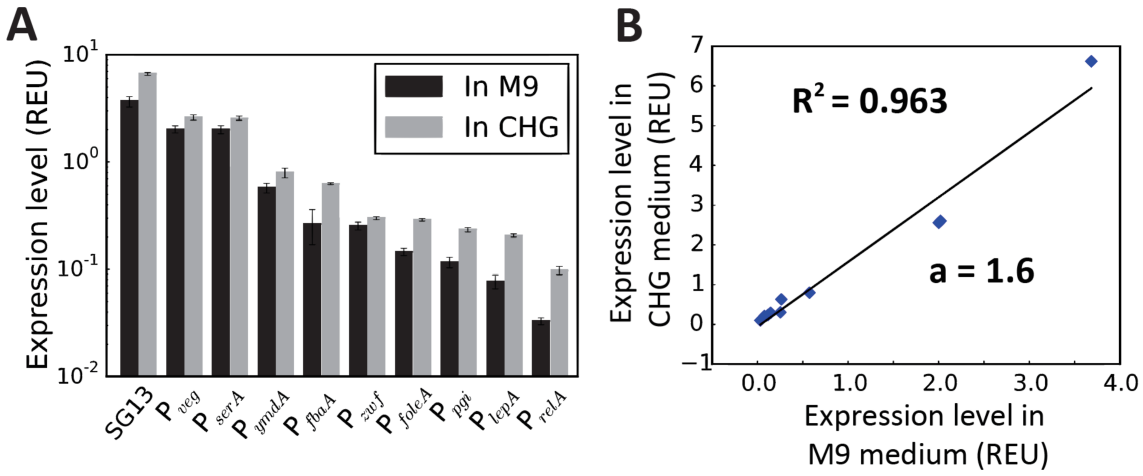
(E) (F) (G): Expression level in exponential phase over expression level in stationary phase in relative expression units for promoters and RBSs sets **(E)**, RBS libraries **(F)** and promoter libraries **(G)**. Linear correlation were performed for each construction sets and a coefficient of determination between 0.83 and 0.999 were found.

Figure S8: Measurement of parts activities in 2 different laboratories.



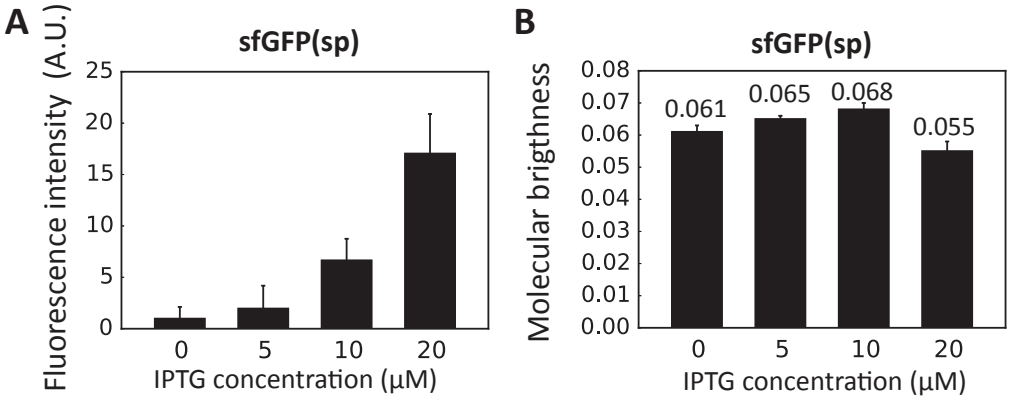
(A) Expression levels in relative expression unit correspond to GFP/OD for 3 kinetic experiments on a plate reader (3 replicates per experiment) in CHG medium (more details in methods). Experiments were performed in parallel in 2 different laboratories with 2 different experimenters. Error-bars correspond to standard deviation over the three experiments. SG13 correspond to the P_{veg} promoter without the +1/+8 part **(B)** Correlation of expression level between both laboratories in REU with coefficient of determination of 0.9995. Full library measurements data are available in supplementary 7.

Figure S9: Correlation of gene expression levels between two medium conditions: M9 and CHG.



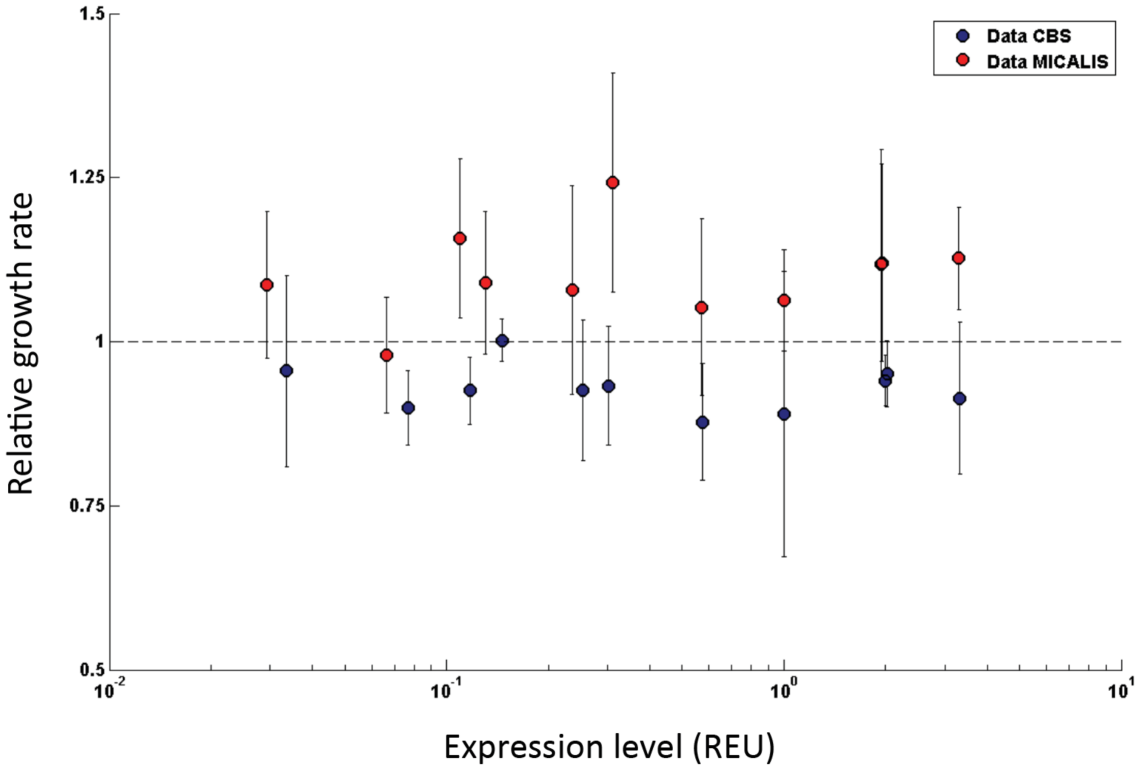
(A) Expression level in REU of the basic promoter set measured by flow-cytometer in exponential phase in M9 (black bars – Figure 2) and measured on a plate-reader in CHG media (grey bars – Figure S8). SG13 correspond to the P_{veg} promoter without the +1/+8 part **(B)** Correlation between experiments on M9 using flow-cytometer and on CHG using plate-reader. Linear correlation with coefficient of determination of 0.963 and director coefficient of 1.6.

Figure S10: Characterisation of sfGFP(sp) using 2-photon microscope and number and brightness method.



(A) (B) Characterisation of the molecular brightness of the sfGFP(sp) at different concentrations of IPTG using 2p sN&B method. sfGFP(sp) expression was induced at different levels using pHyperspank promoter and IPTG concentration from 0 to 20 µM. Fluorescence intensity **(A)**, number of GFP per excitation volume and molecular brightness **(B)** were determined at each IPTG concentration. Error bars correspond to cell-to-cell variation (experiments were performed once).

Figure S11: Growth rate for B. subtilis strains with various GFP expression level.



Relative growth rate correspond to growth rate of strains with a specific expression level in REU over growth rate of the negative control. Data represented correspond to the characterisation of 10 strains with different promoters in 2 different laboratories: CBS and Micalis (see Fig S8). Error-bars correspond to standard deviation over the three experiments.

DNA sequences

Expression Operating Unit (example with Pveg.R0.sfGFP(sp))

SpO-Sp3-B0014-Sp4-**PVEG**-TSS-**NheI**-**R0**-sfGFP(sp)-Sp5-**SphI**-Sp5'-B0015-Sp6-**SpN**

CTCGGATACCCTTACTCTGTTGAAAACGAATAGATAGGTTAAGGAACGGTTATTTCTGCGTAGATCTATCTTACACAGCA
TCACACTGGCTCACCTTCGGGTGGGCCTTTCTGCGTTTATATACTAGAGAGAGAATATAAAAAGCCAGATTATTAATCCG
GCTTTTTTTATTATTTAGGCAACTGAAACGATTCGGATCCTGTATTACTATTCTTA**AATTTTGTCAAAATAATTTTATTGA**
CAACGTCTTATTAACGTTGATACCGGTTAAATTTTATTTGACAAAAATGGGCTCGTGTTGTACAATAAATGTGGAGAAAA
GCTAGCGATTAACTAATAAGGAGGACAAAC**ATG**TCAAAAGGAGAAGAACTTTTTACAGGTGTAGTACCTATCTTGGTTGA
ATTGGATGGTGATGTTAACGGTCACAAATTTTCTGTACGTGGTGAAGGTGAAGGTGATGCAACTAACGGTAAATTGACAC
TTAAATTCATTTGTACAACTGGAAAACTTCCTGTTCCTTGGCCTACTCTTGTTACAACATTGACATATGGAGTACAATGT
TTTTCACGTTATCCTGATCATATGAAACGTCACGATTTTTTTAAATCTGCTATGCCAGAAGGTTATGTACAAGAACGTAC
AATTTCATTTAAAGATGACGGAACATATAAAACACGTGCTGAAGTAAAATTCGAAGGTGACACTCTTGTTAATCGTATCG
AATTGAAAGGAATCGATTTCAAAGAAGATGGTAACATTTTGGGACACAAACTTGAATACAACTTCAACTCTCATAATGTT
TATATCACAGCTGACAAACAAAAAAACGGTATTAAAGCTAATTTTAAAATTCGTCACAATGTTGAAGATGGATCTGTTCA
ATTGGCTGATCATTATCAACAAAATACACCAATCGGAGACGGACCAGTATTGCTTCCAGATAACCACTACCTTTCTACTC
AATCAGTTCTTTCAAAAGATCCTAACGAAAAACGTGACCATATGGTACTTCTTGAATTTGTTACAGCAGCAGGTATCACT
CACGGTATGGACGAACTTTATAAATAAACTTTATCTGAGAATAGTCAATCTTCGGAAATCCCAGGTGGCATGCTAAAAGT
CTCGTAAAGCGTTCTATCAATAACCCGTTGGTGCCAGGCATCAAATAAAACGAAAGGCTCAGTCGAAAGACTGGGCCTTT
CGTTTTATCTGTTGTTTGTCGGTGAACGCTCTCTACTAGAGTCACACTGGCTCACCTTCGGGTGGGCCTTTCTGCGTTTA
TACCGTCTCAGAATCGGCCGTGAACAATAAAATAGTTTCGGT**ATTATTGACCACTTCCGAGTAGAATCGTGCTTCAGTAA**
GA

Phyp.R0.sfGFP(sp).LacI_operon

SpO-Sp3-B0014-Sp4-**PHYP**-**NheI**-**R0**-sfGFP(sp)-Sp5-**SphI**-Sp5'-B0015-Sp6-LacIoperon-Sp7-**SpN**

CTCGGATACCCTTACTCTGTTGAAAACGAATAGATAGGTTAAGGAACGGTTATTTCTGCGTAGATCTATCTTACACAGCA
TCACACTGGCTCACCTTCGGGTGGGCCTTTCTGCGTTTATATACTAGAGAGAGAATATAAAAAGCCAGATTATTAATCCG
GCTTTTTTTATTATTTAGGCAACTGAAACGATTCGGATCCTGTATTACTATTCTTA**CTCGAGGGTAAATGTGAGCACTCAC**
AATTCATTTTGCAAAAGTTGTTGACTTTATCTACAAGGTGTGGCATAATGTGTGTAATTGTGAGCGGATAACAATTGCTA
GCGATTAACTAATAAGGAGGACAAAC**ATG**TCAGAACTAATCAAAGAGAATATGCACATGAAGCTGTACATGGAAGGAACG
GTAAACAATCATCATTTCAAATGTACAAGCGAGGGTGAGGGGAAGCCTTATGAAGGGACACAAACCATGCGGATTAAAGC
AGTCGAGGGCGGACCCCTTCCGTTTGCCTTCGATATCTTGGCTACGAGCTTTATGTATGGGTCGAAAACATTTATCAATC
ACACGCAGGGGATTCCAGACTTTTTCAAACAAAGTTTTCCGGAAGGCTTTACGTGGGAACGTGTGACCACGTATGAAGAT
GGCGGCGTCTTAACAGCTACACAAGATACATCTTTACAAGACGGATGCTTGATATACAACGTTAAGATTCGCGGTGTTAA
CTTTCCGTCAAACGGACCTGTTATGCAGAAGAAAACCCTGGGCTGGGAAGCGTCAACAGAAACACTCTATCCAGCCGACG
GTGGACTTGAGGGCCGTGCCGATATGGCTCTTAAACTCGTGGGCGGTGGCCATCTGATTTGCAATCTTAAAACTACTTAT
CGGTCCAAAAAGCCGGCGAAGAATTTGAAAATGCCTGGAGTATACTACGTTGATAGACGATTAGAAAGGATTAAAGAAGC
AGACAAAGAAACTTATGTAGAGCAGCATGAAGTCGCAGTGGCGAGATATTGTGATTTACCGTCTAAACTGGGACATCGCT
AAACTTTATCTGAGAATAGTCAATCTTCGGAAATCCCAGGTG**GCATGC**TAAAAGTCTCGTAAAGCGTTCTATCAATAACC
CGTTGGTGCCAGGCATCAAATAAAACGAAAGGCTCAGTCGAAAGACTGGGCCTTTCGTTTTATCTGTTGTTTGTCGGTGA
ACGCTCTCTACTAGAGTCACACTGGCTCACCTTCGGGTGGGCCTTTCTGCGTTTATACCGTCTCAGAATCGGCCGTGAAC
AATAAAATAGTTTCGGTTTGCATTTAAATCTTACATATGTAATACTTTCAAAGACTACATTTGTAAGATTTGATGTTTGA
GTCGGCTGAAAGATCGTACGTACCAATTATTGTTTCGTGATTGTTCAAGCCATAACACTGTAGGGATAGTGGAAAGAGTG
CTTCATCTGGTTACGATCAATCAAATATTCAAACGGAGGGAGACGATTTTGATGAAACCAGTAACGTTATACGATGTCGC

AGAGTATGCCGGTGTCTCTTATCAGACCGTTTCCCGCGTGGTGAACCAGGCCAGCCACGTTTCTGCGAAAACGCGGGAAA
AAGTGGAAGCGGCGATGGCGGAGCTGAATTACATTCCCAACCGCGTGGCACAACAACTGGCGGGCAAACAGTCGTTGCTG
ATTGGCGTTGCCACCTCCAGTCTGGCCCTGCACGCGCCGTCGCAAATTGTCGCGGCGATTAAATCTCGCGCCGATCAACT
GGGTGCCAGCGTGGTGGTGTCGATGGTAGAACGAAGCGGCGTCGAAGCCTGTAAAACGGCGGTGCACAATCTTCTCGCGC
AACGCGTCAGTGGGCTGATCATTAACTATCCGCTGGATGACCAGGATGCCATTGCTGTGGAAGCTGCCTGCACTAATGTT
CCGGCGTTATTTCTTGATGTCTCTGACCAGACACCCATCAACAGTATTATTTTCTCCCATGAAGACGGTACGCGACTGGG
CGTGGAGCATCTGGTCGCATTGGGTCACCAGCAAATCGCGCTGTTAGCGGGCCCATTAAGTTCTGTCTCGGCGCGTCTGC
GTCTGGCTGGCTGGCATAAATATCTCACTCGCAATCAAATTCAGCCGATAGCGGAACGGGAAGGCGACTGGAGTGCCATG
TCCGGTTTTCAACAAACCATGCAAATGCTGAATGAGGGCATCGTTCCCACTGCGATGCTGGTTGCCAACGATCAGATGGC
GCTGGGCGCAATGCGCGCCATTACCGAGTCCGGGCTGCGCGTTGGTGCGGATATCTCGGTAGTGGGATACGACGATACCG
AAGACAGCTCATGTTATATCCCGCCGTTAACCACCATCAAACAGGATTTTCGCCTGCTGGGGCAAACCAGCGTGGACCGC
TTGCTGCAACTCTCTCAGGGCCAGGCGGTGAAGGGCAATCAGCTGTTGCCCGTCTCACTGGTGAAAAGAAAAACCACCCT
GGCGCCCAATACGCAAACCGCCTCTCCCCGCGCGTTGGCCGATTCATTAATGCAGCTGGCACGACAGGTTTCCCGACTGG
AAAGCGGGCAGTGATAATAAAAGGTCCCGTCTGAACTTACTGTGAATTCGACTA**ATTATTGACCACTTCCGAGTAGAATC**
GTGCTTCAGTAAGA

mKate2(Bs)

ATGTCAGAACTAATCAAAGAGAATATGCACATGAAGCTGTACATGGAAGGAACGGTAAACAATCATCATTTCAAATGT
ACAAGCGAGGGTGAGGGGAAGCCTTATGAAGGGACACAAACCATGCGGATTAAAGCAGTCGAGGGCGGACCCCTT
CCGTTTGCCTTCGATATCTTGGCTACGAGCTTTATGTATGGGTCGAAAACATTTATCAATCACACGCAGGGGATTCCA
GACTTTTTCAAACAAAGTTTTCCGGAAGGCTTTACGTGGGAACGTGTGACCACGTATGAAGATGGCGGCGTCTTAAC
AGCTACACAAGATACATCTTTACAAGACGGATGCTTGATATACAACGTTAAGATTCGCGGTGTTAACTTTCCGTCAAA
CGGACCTGTTATGCAGAAGAAAACCCTGGGCTGGGAAGCGTCAACAGAAACACTCTATCCAGCCGACGGTGGACTT
GAGGGCCGTGCCGATATGGCTCTTAAACTCGTGGGCGGTGGCCATCTGATTTGCAATCTTAAAACTACTTATCGGTC
CAAAAAGCCGGCGAAGAATTTGAAAATGCCTGGAGTATACTACGTTGATAGACGATTAGAAAGGATTAAAGAAGCAG
ACAAAGAAACTTATGTAGAGCAGCATGAAGTCGCAGTGGCGAGATATTGTGATTTACCGTCTAAACTGGGACATCGC
TAA

BCDs

BCD-4

RBS0-first start codon-RBS4-second start codon

GGGCCCAAGTTCACTTAAGATTAACTAATAAGGAGGACAACAACAATGAAAGCAATTTTCGTACTGAAtgacatgaaagg
aagtatttgatAATG

BCD-7

RBS0-first start codon-RBS7-second start codon

GGGCCCAAGTTCACTTAAGATTAACTAATAAGGAGGACAACAACAATGAAAGCAATTTTCGTACTGAAggtgggaaggag
gaactactAATG

Spacer sequences

Spacer names	Spacer sequences
sp0	CTCGGATACCCTTACTCTGTTGAAAACGAATAGATAGGTT
sp1	TGCTCGTAGTTTACCACGGATACAGACAGTGATAATCTTA
sp2	AGATTACTACTGATAACCACTGTTGATTGGGATACCCGTA
sp3	AAGGAACGGTTATTTCTGCGTAGATCTATCTTACACAGCA
sp4	AGGCAACTGAAACGATTCGGATCCTGTATTACTATTCTTA
sp5	ACTTTATCTGAGAATAGTCAATCTTCGGAAATCCCAGGTG
sp5'	TAAAAGTCTCGTAAAGCGTTCTATCAATAACCCGTTGGTG
sp6	CCGTCTCAGAATCGGCCGTGAACAATAAAATAGTTTCGGT
sp7	TAATAAAAGGTCCCGTCTGAACTTACTGTGAATTCGACTA
spN	ATTATTGACCACTTCCGAGTAGAATCGTGCTTCAGTAAGA

Primer sequences

Primer numbers	Primer sequences
16	gccgcgatttccaatgaggtta
31	caacagagtaagggtatccgagcgatcagaccagtttttaatttgtgtg
32	gagtagaatcgtgcttcagtaagaggcgattttcgttcgtgaatac
33	CTCGGATACCCTTACTCTGTTGAAAAC
34	TCTTACTGAAGCACGATTCTACTCGG
36	TGCTGTGTAAGATAGATCTACGCAG
39	ACTTTATCTGAGAATAGTCAATCTTCGGAAATC
40	CACCTGGGATTTCCGAAGATTGAC
51	GGATCGGAgcatgcTTAAHNAHNAHNAGCAACATTTTGATTAAATGAATTTGTTTTGCCTGCtttataaagttcgtccataccgtgagtg
62	ACGTTGATACCGGTTAAATTTTATNNNNNNAAAATGGGCTCGTGTTGNNNNNNaaatgtgctagcgattaactaataaggagg
63	ACGTTGATACCGGTTAAATTTTATNNNACAAAAATGGGCTCGTGTTGTATAATaaatgtgctagcgattaactaataaggagg
64	ACGTTGATACCGGTTAAATTTTATTTGACAAAAATGGGCTCGTGTTGNNNAATaaatgtgctagcgattaactaataaggagg
71	cgttttcaacagagtaagggtatccgag
72	attattgaccacttccgagtagaatcgtg
109	gattaactaataaggaggacaaacatgtc
124	TTTATAAAGTTCGTCCATACCGTGAGTGATACC
137	ggttaaatttttatttgacaaaaatgggctcgtgttgtacaataaatgtggagaaaagctagcgattaactaataaggaggacaaac
139	ggttaaatttttatttgacaaaaatgggctcgtgttgtacaataaatgtggagaaaagctagcgattaactaataaggaggacaaac
140	ggctcgtgttgtacaataaatgtggagaaaagctagcggtgggaaggaggggttcgacatgtcaaaaggagaagaacttttacagg
141	ggctcgtgttgtacaataaatgtggagaaaagctagcggtgggaaggaggaactactatgtcaaaaggagaagaacttttacagg
142	ggctcgtgttgtacaataaatgtggagaaaagctagcggtgggaaggaggacattcgacatgtcaaaaggagaagaacttttacagg
143	ggctcgtgttgtacaataaatgtggagaaaagctagcaaaggaggtgatgacatgtcaaaaggagaagaac

ttttacagg

144 ggctcgtgttgtacaataaatgtggagaaaagctagcgctcttaaggaggattttagaatgtcaaaaggag
aagaacttttacagg

145 ggctcgtgttgtacaataaatgtggagaaaagctagctgacatgaaaggaagtatttgaaaatgtcaaaag
gagaagaacttttacagg

146 gctagcttttctccacatttattgtac

160 gattaactaataaggaggacaaacatgtcagaactaatc

161 gattagttctgacatgtttgtcctccttattagttaatc

162 catttattgtacaacacgagc

206 ggctcgtgttgtacaataaatgtggagaaaagctagcggtgggaaggaggtgatccaatgtcagaactaat
caaagagaatatgcac

207 ggctcgtgttgtacaataaatgtggagaaaagctagcggtgggaaggaggggttcgacatgtcagaacta
atcaaagagaatatgcac

208 ggctcgtgttgtacaataaatgtggagaaaagctagcggtgggaaggaggaactactatgtcagaactaat
caaagagaatatgcac

209 ggctcgtgttgtacaataaatgtggagaaaagctagcggtgggaaggaggacattcgacatgtcagaacta
atcaaagagaatatgcac

210 ggctcgtgttgtacaataaatgtggagaaaagctagcaaaggaggtgatgacatgtcagaactaatcaaag
agaatatgcac

211 ggctcgtgttgtacaataaatgtggagaaaagctagcgctcttaaggaggattttagaatgtcagaactaa
tcaaagagaatatgcac

212 ggctcgtgttgtacaataaatgtggagaaaagctagctgacatgaaaggaagtatttgaaaatgtcagaac
taatcaaagagaatatgc

213 ACGTTGATaccggttaaattttatttgacaaaaatgggctcgtgttgNNNaataaatgtggagaaaagcta
gcgattaac

214 CGTTGATggatcctgtattactattcttaactgcgtcaatacacgttgacactcttttgagaatatgtNNN
attatcagggagaaaagctagcgattaac

215 CGTTGATggatcctgtattactattcttagttaagatggcaagcttgacaagtatttccgacacattNNNa
atgaagttggagaaaagctagcgattaac

220 GTGATCCAgctagcgattaactaataaNNNNNNcaaacatgtcaaaaggagaagaacttttacagg

221 GTGATCCAgctagcGGTGGAANNNNNNTGATGACatgtcaaaaggagaagaacttttacagg

222 GTGATCCAgctagcgctcttaNNNNNNattttagaatgtcaaaaggagaagaacttttacagg

225 ggctcgtgttgtacaataaatgtggagaaaagctagcGGTGGAAAGGAGGTGATGACatgtcaaaaggaga
agaacttttacagg

238 ggctcgtgttgtacaataaatgtggagaaaagctagcgattaactaataaggaggacaaacatgtcagaac
taatcaaagagaatatgc

341 CGTTGATggatcctgtattactattcttaactgcgtcaatacacgttgacactcttttgNNNNNNtgttaa
attatcagggagaaaagct

342 CGTTGATggatcctgtattactattcttagttaagatggcaagcttgacaagtatttcNNNNNNatttaca
atgaagttggagaaaagct

343 aggcaactgaaacgattcggatcctgtattactattcttaggagaaaagctagcgattaactaataaggag
gac

423 ctgtaaaaagttcttctccttttgacatgctagcacatttattgtacaacacg

424 cgtgttgtacaataaatgtgctagcatgtcaaaaggagaagaacttttacag

Gblock sequences

Gblock names	Gblock sequences
ssrA_LAA	CAGCAGCAGGTATCACTCACGGTATGGACGAACTTTATAAAGCAGGTAAGACTAATTCATTTAATC AAAATGTTGCTCTTGCAGCATAAACTTTATCTGAGAATAGTCAATCTTCGGAAATCCCAGGTG
ssrA_LDD	CAGCAGCAGGTATCACTCACGGTATGGACGAACTTTATAAAGCAGGTAAGACTAATTCATTTAATC AAAATGTTGCTCTTGATGATTAAACTTTATCTGAGAATAGTCAATCTTCGGAAATCCCAGGTG
ssrA_AAV	CAGCAGCAGGTATCACTCACGGTATGGACGAACTTTATAAAGCAGGTAAGACTAATTCATTTAATC AAAATGTTGCTGCTGCTGTTTAAACTTTATCTGAGAATAGTCAATCTTCGGAAATCCCAGGTG
ssrA_ASV	CAGCAGCAGGTATCACTCACGGTATGGACGAACTTTATAAAGCAGGTAAGACTAATTCATTTAATC AAAATGTTGCTGCTAGTGTTTAAACTTTATCTGAGAATAGTCAATCTTCGGAAATCCCAGGTG
ssrA_LVA	CAGCAGCAGGTATCACTCACGGTATGGACGAACTTTATAAAGCAGGTAAGACTAATTCATTTAATC AAAATGTTGCTTTAGTTGCTTAAACTTTATCTGAGAATAGTCAATCTTCGGAAATCCCAGGTG
Gblock_fbaA	ctcggataccttactctgttgaaaacgaatagataggttaaggaacggttatttctgcgtagatc tatcttacacagcatcacactggctcaccttcgggtgggcctttctgcgtttatatactagagaga gaatataaaaagccagattattaatccggctttttttattatttaggcaactgaaacgattcggatc ctgtattactattcttaaatcatgtcattatgttgccgatttgtcgaaaagttggtatcctagtta tggagaaaagctagcgattaactaataaggaggacaaacatgtcaaaaggagaagaacttttttaca ggtgtagtacctatcttggttg
Gblock_zwf	ctcggataccttactctgttgaaaacgaatagataggttaaggaacggttatttctgcgtagatc tatcttacacagcatcacactggctcaccttcgggtgggcctttctgcgtttatatactagagaga gaatataaaaagccagattattaatccggctttttttattatttaggcaactgaaacgattcggatc ctgtattactattcttaaaaagggcttaaatgtttgctttcgttgaattttagatttaaaatgaag gggagaaaagctagcgattaactaataaggaggacaaacatgtcaaaaggagaagaacttttttaca ggtgtagtacctatcttggttg
Gblock_ymda	ctcggataccttactctgttgaaaacgaatagataggttaaggaacggttatttctgcgtagatc tatcttacacagcatcacactggctcaccttcgggtgggcctttctgcgtttatatactagagaga gaatataaaaagccagattattaatccggctttttttattatttaggcaactgaaacgattcggatc ctgtattactattcttagttaagatggcaagcttgacaagtatttccgacacatttacaatgaagt tggagaaaagctagcgattaactaataaggaggacaaacatgtcaaaaggagaagaacttttttaca ggtgtagtacctatcttggttg
Gblock_serA	ctcggataccttactctgttgaaaacgaatagataggttaaggaacggttatttctgcgtagatc tatcttacacagcatcacactggctcaccttcgggtgggcctttctgcgtttatatactagagaga gaatataaaaagccagattattaatccggctttttttattatttaggcaactgaaacgattcggatc ctgtattactattcttaactgcgtcaatacacgttgacactcttttgagaatatgttaaattatca gggagaaaagctagcgattaactaataaggaggacaaacatgtcaaaaggagaagaacttttttaca ggtgtagtacctatcttggttg
Gblock_pgi	ctcggataccttactctgttgaaaacgaatagataggttaaggaacggttatttctgcgtagatc tatcttacacagcatcacactggctcaccttcgggtgggcctttctgcgtttatatactagagaga gaatataaaaagccagattattaatccggctttttttattatttaggcaactgaaacgattcggatc ctgtattactattcttacctttcttcttgacttgatttcacagataagttcatataaagtgaaaga tggagaaaagctagcgattaactaataaggaggacaaacatgtcaaaaggagaagaacttttttaca ggtgtagtacctatcttggttg
Gblock_relA	ctcggataccttactctgttgaaaacgaatagataggttaaggaacggttatttctgcgtagatc tatcttacacagcatcacactggctcaccttcgggtgggcctttctgcgtttatatactagagaga gaatataaaaagccagattattaatccggctttttttattatttaggcaactgaaacgattcggatc ctgtattactattcttaacttctgctctttacatctttcgtttttttcttgataataaactacaat aggagaaaagctagcgattaactaataaggaggacaaacatgtcaaaaggagaagaacttttttaca ggtgtagtacctatcttggttg
Gblock_folEA	ctcggataccttactctgttgaaaacgaatagataggttaaggaacggttatttctgcgtagatc tatcttacacagcatcacactggctcaccttcgggtgggcctttctgcgtttatatactagagaga gaatataaaaagccagattattaatccggctttttttattatttaggcaactgaaacgattcggatc ctgtattactattcttagggcattcactttgcttttagcggggcatatgtgctagaatcgaaatta

```
                aggagaaaagctagcgattaactaataaggaggacaaacatgtcaaaaggagaagaacttttttaca
                ggtgtagtacctatcttggttg
Gblock_lepA     ctcggatacccttactctgttgaaaacgaatagataggttaaggaacggttatttctgcgtagatc
                tatcttacacagcatcacactggctcaccttcgggtgggcctttctgcgtttatatactagagaga
                gaatataaaaagccagattattaatccggctttttttattatttaggcaactgaaacgattcggatc
                ctgtattactattcttaagtcaatgtatgaatggatacgggatatgaatcaataagtacgtgaaag
                agaaaagcaacccagatatgatagggaacttttctctttcttgttttacattgaatctttacaatc
                ctattgatataatctaagctagtgtattttgcgtttaatagtggagaaaagctagcgattaactaa
                taaggaggacaaacatgtcaaaaggagaagaacttttttacaggtgtagtacctatcttggttg
```

Annex D

Supplementary Data: An automated design framework for multicellular recombinase logic.

Supplementary Information: An automated design framework for multicellular recombinase logic.

Sarah Guiziou[1], Federico Ulliana[2], Violaine Moreau[1], Michel Leclere[2], and Jerome Bonnet*[1]

[1]Centre de Biochimie Structurale, INSERM U1054, CNRS UMR5048, University of Montpellier, France.
[2]Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM). CNRS UMR 5506, University of Montpellier, France.
*To whom correspondence should be addressed: jerome.bonnet@inserm.fr

These supplementary materials contain:
-Supplementary Figures S1 to S3.
-Supplementary Tables S1.

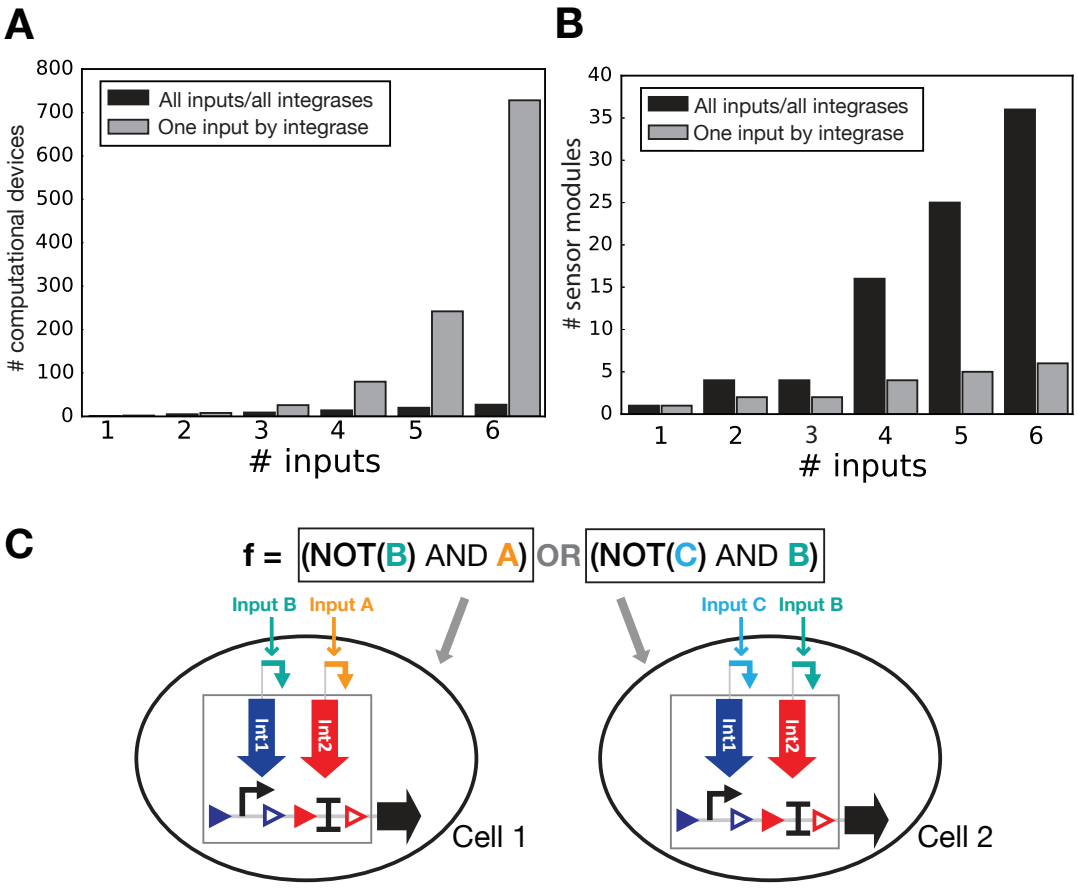


Figure S1: **Reduction of the number of Boolean logic devices by connecting all inputs to all integrases. (A)** Reduction of the number of computational devices needed by connection of all inputs to all integrases. The bar graph represents the number of standard computational devices needed to implement a function responding to a specific number of inputs, with the black bars for connection of all inputs to all integrases and the grey bars for connection of one input to one integrase (see methods for equation). **(B)** Number of sensor modules needed using all-input/all-integrase design or one input by integrase design. If only one device per symmetric function is implemented, all combinations of inputs with integrases have to be built to implement all logic sub-functions. The number of sensor modules with this design strategy is higher than in the one input by integrase design. The bar graph represents the number of sensor modules needed in function of the input number, for all-input/all-integrase design (black bars) and one input by integrase design (grey bars). By comparing A and B, it is clear that the total number of component needed is greatly in favor of the all-input/all-integrase design. **C** - Example of Boolean logic implementation based on two cells using the same computational devices and different input-integrase connections.

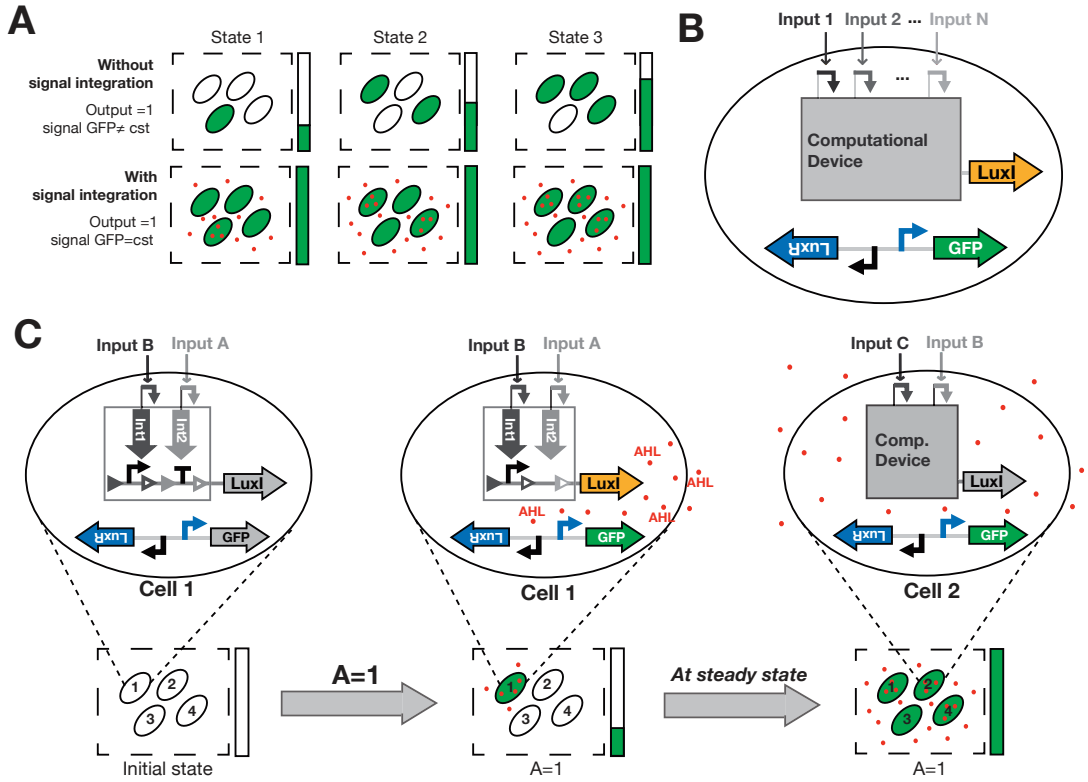


Figure S2: **Use of cell-cell communication to obtain a constant output signal between states. A** - The integration of the output signal is required to obtain a uniform output in all ON states. In our multicellular design, the output is considered equal to one if at least one cellular computing unit is ON. Therefore, the expression level of the output gene will be different if one or several units are ON. For applications that require a constant output level, integration of the output signal might be performed using cell-cell communication. If one of the strains is ON, it produces an AHL molecule that is detected by the other strains, which subsequently turn ON such that in all ON states of the program the output level is constant. **B** - Implementation of cell-cell communication to integrate output signals. The output gene of the computational device is a gene producing an AHL molecule (for example LuxI), and the output gene (here GFP) is connected to a promoter inducible by AHL. **C** - Example of the behavior of a cellular computing unit with a signal integration system. In the initial state for this specific strain, the output gene of the computing device (LuxI) is OFF as for all other strains. Then, with the presence of the input A, the terminator is excised, LuxI is expressed, and AHL is produced. GFP will be expressed in this strain, and by diffusion of AHL all strains will produce GFP.

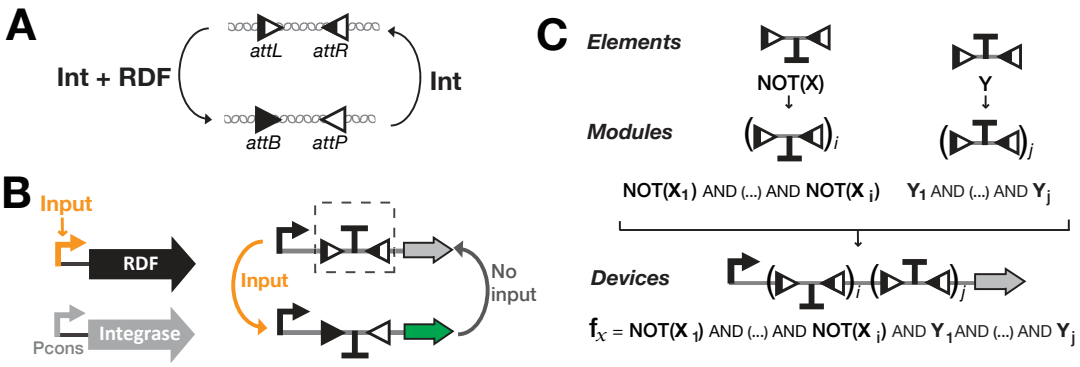


Figure S3: **Hierarchical composition framework for synchronous Boolean logic using integrases and recombination directionality factors (RDFs). A** - Reversible inversion of DNA using integrase coupled with RDF. Integrase alone specifically targets *attB* and *attP* sites and does not operate on *attL* and *attR* sites. When sites are oriented in the opposite direction, DNA between sites is inverted and *attL* and *attR* sites are formed. With the additional use of a RDF, the integrase targets specifically *attL* and *attR* sites and inverts DNA between these sites, reverting to *attB* and *attP*. Therefore, using a RDF enables the implementation of reversible integrase-based DNA switches[14]. **B** - To obtain a synchronous IDENTITY function, the integrase sites *attL* and *attR* are placed in inverted orientation around an asymmetric terminator. The terminator blocks the flow of RNA polymerase, and the output gene is not expressed. The integrase is constitutively expressed in all states. When the input is present, RDF is expressed and the terminator inverted. As the terminator is asymmetric, the output gene is expressed[10,30]. **C** - Hierarchical composition of synchronous elements. NOT- and ID-elements are composed with *attL* and *attR* sites in inversion mode flanking an asymmetric transcriptional terminator. For the NOT-element, the terminator is in the OFF position and for the ID-element in the ON position. ID- and NOT-modules are both composed in series between the promoter and the output gene and compute, respectively, the conjunction of IDENTITY functions and NOT functions. The device is then expandable by addition of elements in series to all logic functions based on the conjunction of NOT and IDENTITY functions.

# Strains	# 3-input functions	# 4-input functions
1	26	80
2	130	1804
3	88	13472
4	10	28904
5	–	17032
6	–	3704
7	–	512
8	–	26

Table 1: **Proportion of Boolean functions implementable with a specific number of strains for 3 and 4 inputs.** This table was obtained by systematic generation of the biological design of all 3 and 4-input Boolean functions using our python software.

Annex E

Supplementary Data of History-dependent programs: Cell History Tracker

---

By interlacing target sites for different recombinases, recombination reactions can be made dependent on one other. Using this concept, researchers started to implement genetic devices tracking the order of occurrence of signals, as well as history-dependent gene expression programs. We found that a basic history-dependent motif could be repeatedly distributed into different cells to straightforwardly implement all input event-order trackers using a multicellular consortia (Figure S1, Table S1). The state of the tracker could be addressed experimentally via multiplexed next-generation sequencing.

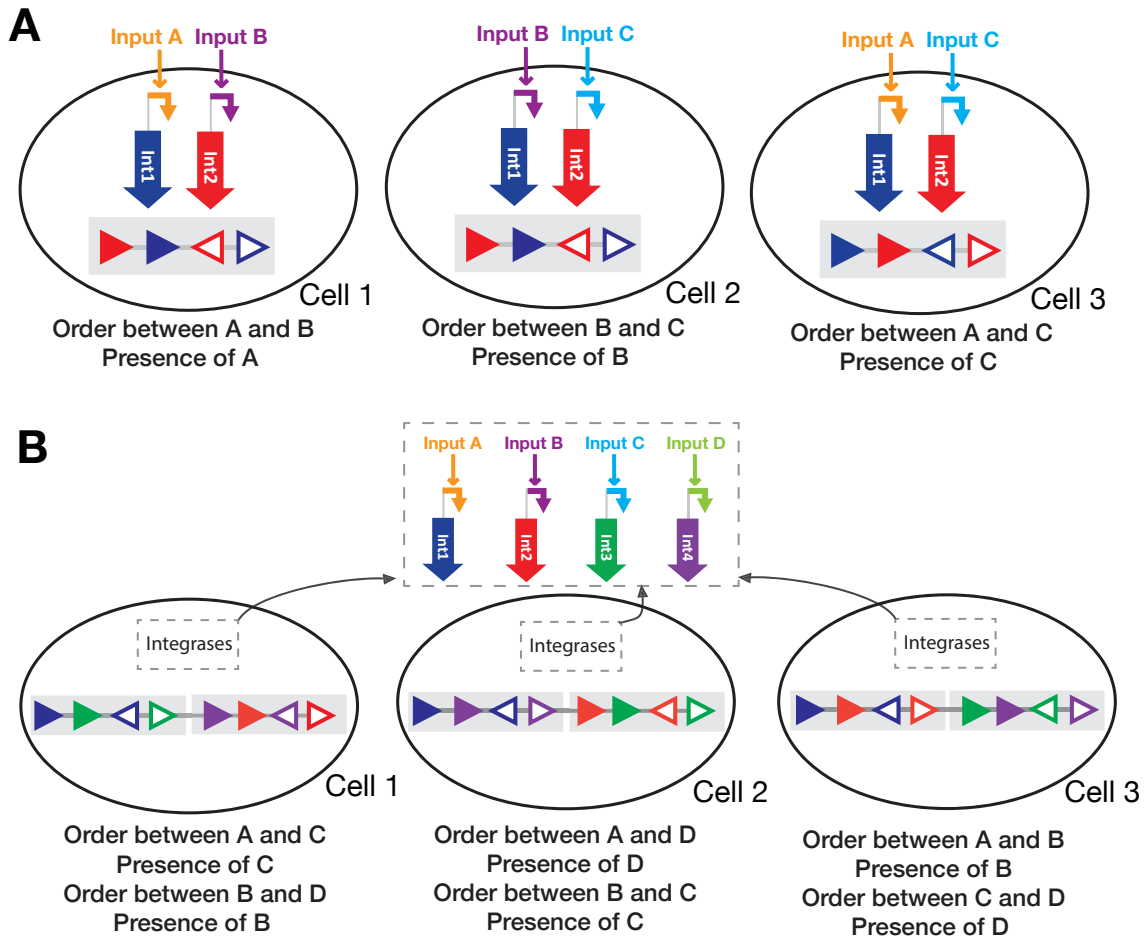


Figure E.1: **Examples of designs for 3- and 4-input event-order trackers using multicellular consortia. A -** 3-input event-order tracker design. The design is based on the repetition of a DNA tracker module composed of two interlaced pairs of integrase sites that allow the determination of the order of occurrence of 2-inputs and the presence of one of them. For 3-inputs, three DNA tracker modules are needed. To limit the number of integrases and strains needed, the system is implemented in three cells with one tracker module and two integrases per cell. The same two integrases can be used in all cells to reduce the number of different integrases required. Consequently, control signals and integrases are connected differently. **B -** 4-input event-order tracker design. For 4-inputs, six DNA tracker modules are needed. The system is implemented in three cells with two tracker modules and four integrases per cell. The same input-integrase connections are used in all cells.

# Inputs	*# Cell types*	*# Integrases per Cell type*
1	1	1
2	2	2
3	3	2
4	3	4
5	5	4
6	5	6
7	7	6
8	7	8
9	9	8
10	9	10

Table E.1: **Metrics from 1- to 10-input event-order trackers using a multicellular consortia.** This table lists the number of cells and integrases per cell needed for implementation of event-order trackers with from 1 to 10 inputs.

Annex F

DNA sequences of parts, primers, fragments.

F.1 DNA sequences of parts

Integrase	Site	DNA sequence
Bxb1	attB	TCGGCCGGCTTGTCGACGACGGCGGTCTCCGTCGTCAGGATCATC CGGGC
Bxb1	attP	TCGTGGTTTGTCTGGTCAACCACCGCGGTCTCAGTGGTGTACGGT ACAAACCC
Bxb1	attL	TCGGCCGGCTTGTCGACGACGGCGGTCTCAGTGGTGTACGGTACA AACCC
Bxb1	attR	TCGTGGTTTGTCTGGTCAACCACCGCGGTCTCCGTCGTCAGGATC ATCCGGGC
Tp901	attB	ATGCCAACACAATTAACATCTCAATCAAGGTAAATGCTTTTTGCT TTTTTTGC
Tp901	attP	GCGAGTTTTTATTTCGTTTATTTCAATTAAGGTAACTAAAAAACT CCTTT
Tp901	attL	ATGCCAACACAATTAACATCTCAATTAAGGTAACTAAAAAACTCC TTT
Tp901	attR	GCGAGTTTTTATTTCGTTTATTTCAATCAAGGTAAATGCTTTTTG CTTTTTTTGC
Int5	attB	gagcgccggatcagggagtggacggcctgggagcgctacacgctgtggctgcggtcggtgc
Int5	attP	ccctaatacgcaagtcgataactctcctgggagcgttgacaacttgcgcaccctgatctg
Int5	attL	gagcgccggatcagggagtggacggcctgggagcgttgacaacttgcgcaccctgatctg
Int5	attR	ccctaatacgcaagtcgataactctcctgggagcgctacacgctgtggctgcggtcggtgc
Int7	attB	agacgagaaacgttccgtccgtctgggtcagttgggcaaagttgatgaccgggtcgtccgtt
Int7	attP	gtgttataaacctgtgtgagagttaagtttacatgcctaaccttaacttttacgcaggttcagct t
Int7	attL	agacgagaaacgttccgtccgtctgggtcagttgcctaaccttaacttttacgcaggttcagct t
Int7	attR	gtgttataaacctgtgtgagagttaagtttacatgggcaaagttgatgaccgggtcgtccgtt
Int3	attB	gtttgtaaaggagactgataatggcatgtacaactatactcgtcggtaaaaaggcatcttat
Int3	attP	atggataaaaaaatacagcgtttttcatgtacaactatactagttgtagtgcctaaataatgctt
Int4	attB	ttccaaagagcgcccaacgcgacctgaaatttgaataagactgctgcttgtgtaaaggcgatgatt
Int4	attP	caaaaattacaaagttttcaacccttgatttgaattagcggtcaaataatttgtaattcgttt

Table F.1: Integrase site sequences.

Integrase	DNA sequence
Bxb1	gtgagagccctggtagtcatccgcctgtcccgcgtcaccgatgctacgacttcaccggagcgtcagctggagtcttgcca gcagctctgcgcccagcgcggctgggacgtcgtcggggtagcggaggatctggacgtctccggggcggtcgatccgttc gaccggaagcgcagaccgaacctggcccggtggcttgcgttcgaggagcaaccgtttgacgtgatcgtggcgtaccggg tagatcggttgacccgatcgatccggcatcttcagcagctggtccactgggccgaggaccacaagaagctggtcgtctcc gcgaccgaagcgcacttcgatacgacgacgccgtttgcggcggtcgtcatcgcgcttatgggaacggtggcgcagatgg aattagaagcgatcaaagagcggaaccgttcggctgcgcatttcaatatccgcgccgggaaataccgagggtccctgcc gccgtggggatacctgcctacgcgcgtggacggggagtggcgactggtgccggaccctgtgcagcgagagcgcatcctc gaggtgtatcaccgcgtcgtcgacaaccacgagccgctgcatctggtggcccacgacctgaaccggcgtggtgtcctgtc gccgaaggactacttcgcgcagctgcaaggccgcgagccgcagggccgggagtggtcggctaccgcgctgaagcgatc gatgatctccgaggcgatgctcgggtacgcgactctgaacggtaagaccgtccgagacgacgacggagccccgctggtg cgggctgagccgatcctgacccgtgagcagctggaggcgctgcgcgccgagctcgtgaagacctcccgggcgaagcccg cggtgtctaccccgtcgctgctgctgcgggtgttgttctgcgcggtgtgcggggagcccgcgtacaagttcgccgggggga ggacgtaagcacccgcgctaccgctgccgctcgatggggttcccgaagcactgcgggaacggcacggtggcgatggccg agtgggacgcgttctgcgaggagcaggtactggatctgctcggggacgcggagcgtctggagaaagtctgggtagcgg gctcggactccgcggtcgaactcgcggaggtgaacgcggagctggtggacctgacgtcgctgatcggctccccggccta ccgggcgggctctccgcagcgagaagcactggatgcccgtattgcggcgctggccgcgcggcaagaggagctggaggg cctggaggctcgcccgtctggctgggagtggcgcgagaccgggcagcggttcggggactggtggcgggagcaggacac cgcggcaaagaacacctggcttcggtcgatgaacgttcggctgacgttcgacgtccgcggcgggctgactcgcacgatc gacttcggggatcttcaggagtacgagcagcatctcaggctcggcagcgtggtcgaacggctacacaccgggatgtcgt aa
Tp901	atgactaagaaagtagcaatctatacacgagtatccactactaaccaagcagaggaaggcttctcaattgatgagcaaa ttgaccgtttaacaaaatatgctgaagcaatggggtggcaagtatctgatacttatactgatgctggtttttcaggggcc aaacttgaacgcccagcaatgcaaagattaatcaacgatatcgagaataaagcttttgatacagttcttgtatataagct agaccgcctttcacgtagtgtaagagatactctttatcttgttaaggatgtgttcacaaaaaataaaatagactttatctc gcttaatgaaagtattgatacttcttctgctatgggtagcttgtttctcactattctttctgcaattaatgagtttgaaaga gagaatataaaagaacgcatgactatgggtaaactagggcgagcgaaatctggtaagtctatgatgtggactaagaca gcttttgggtattaccacaacagaaagacaggtatattagaaattgttcctttacaagctacaatagttgaacaaatatt cactgattatttatcaggaatatcacttacaaaattaagagataaactcaatgaatctggacacatcggtaaagatatac cgtggtcttatcgtaccctaagacaaacacttgataatccagtttactgtggttatatcaaatttaaggacagcctatttg aaggtatgcacaaaccaattatcccttatgagacttatttaaaagttcaaaaagagctagaagaaagacaacagcagac ttatgaaagaaataacaaccctagacctttccaagctaaatatatgctgtcagggatggcaaggtgcggttactgtgga gcacctttaaaaattgttcttggccacaaaagaaaagatggaagccgcactatgaaatatcactgtgcaaatagatttcc tcgaaaaacaaaaggaattacagtatataatgacaataaaaagtgtgattcaggaacttatgatttaagtaatttagaa aatactgttattgacaacctgattggatttcaagaaaataatgactccttattgaaaattatcaatggcaacaaccaacc tattcttgatacttcgtcatttaaaaagcaaatttcacagatcgataaaaaaatacaaaagaactctgatttgtacctaa atgattttatcactatggatgagttgaaagatcgtactgattcccttcaggctgagaaaaagctgcttaaagctaagatt agcgaaaataaatttaatgactctactgatgtttttgagttagttaaaactcagttgggctcaattccgattaatgaacta tcatatgataataaaaagaaaatcgtcaacaaccttgtatcaaaggttgatgttactgctgataatgtagatatcatattt aaattccaactcgctaccggttaa

Table F.2: Integrase site sequences.

Integrase	DNA sequence
Int5	atgcctggtatgaccaccgaaaccggtccggatcctgcaggtctgattgacctgttttgtcgtaaaagcaaagcagttaa aagccgtgcaaatggtgcaggtcagcgtcgtaaacaagaaattagcattgcagcacaagaaaccctgggtcgtaaagt tgcagcactgctgggtatgcaggttcgtcatgtttggaaagaagttggtagcgcaagccgttttcgtaaaggtaaagcac gtgatgatcagagcaaagcactgaaagccctggaaagcggtgaagttggtgcactgtggtgttatcgtctggatcgttg ggatcgtggtggtgccggtgcaattctgaaaattatcgaaccggaagatggtatgcctcgtcgtctgctgtttggttggg atgaagataccggtcgtccggttctggatagcaccaataaacgtgatcgcggtgaactgattcgtcgtgcagaagaagc acgcgaagaagcagaaaaactgagcgaacgtgttcgtgataccaaagcacatcagcgtgaaaatggtgaatgggttaa tgcccgtgcaccgtatggtctgcgtgttgttctggttaccgttagtgatgaagagggtgatgaatatgatgaacgtaaac tggcagcagatgatgaagatgcgggtggtcctgatggtctgaccaaagcagaagcagcccgtctggtttttaccctgccg gttaccgatcgtctgagctatgcaggcaccgcacatgcaatgaatacccgtgaaattccgagcccgaccggtggtccgtg gattgcagttaccgtgcgtgatatgattcagaatccggcatatgcgggttggcagaccacaggtcgtcaggatggtaaa cagcgtcgtctgacctttataacggtgaaggtaaacgtgttagcgttatgcatggtcctccgctggtgaccgatgaaga acaagaagccgcaaaagcagccgttaaaggtgaagatggtgttggtgttccgctggatggtagcgatcatgatacccgt cgcaaacatctgctgagcggtcgtatgcgttgtccgggttgtggtggtagctgtagctatagcggtaatggttatcgttgt tggcgtagcagtgtgaaaggtggttgtccggcaccgacctatgttgcacgtaaaagcgttgaagaatatgttgcatttcg ttgggcagcaaaattagcagcaagcgaaccggatgatccgtttgttattgcagttgcagatcgctgggcagcactgaccc atccgcaggcaagcgaagatgaaaagtatgcaaaagccgcagttcgtgaagccgaaaaaaatctgggtcgcctgctgc gtgatcgtcagaatggtgtttatgatggtccggcagaacagttttttgcccctgcatatcaagaagcactgagcaccctg caggcagccaaagatgcagttagcgaaagcagcgcaagcgcagcagttgatgttagctggattgttgatagcagcgatt atgaagaactgtggctgcgtgcaaccccgaccatgcgtaatgcaattattgatacctgcatcgatgaaatttgggttgca aaaggccagcgtggtcgtccgtttgatggtgatgaacgcgttaaaatcaaatgggcagcccgtacctaa
Int7	atgaaagtggccatttatgttcgtgttagcaccgatgaacaggccaaagaaggttttagcattccggcacagcgtgaac gtctgcgtgcattttgtgcaagccagggttgggaaattgtgcaagaatatattgaagaaggttggagcgcaaaagatct ggatcgtccgcagatgcagcgtctgctgaaagatatcaaaaaaggcaacattgatattgtgctggtgtatcgtctggatc gcctgacccgtagcgttctggatctgtatctgctgctgcagacctttgaaaaatacaatgtggcatttcgtagcgccaccg aagtttatgataccagcaccgcaatgggtcgtctgtttattaccctggttgcagcactggcacagtgggaacgtgaaaat ctggcagaacgtgttaaatttggtatcgagcagatgatcgatgaaggtaaaaaaccgggtggtcatagcccgtatggtt acaaatttgataaagacttcaattgcaccattattgaggaagaagcagacgttgttcgtatgatctatcgcatgtattgtg atggttatggctatcgtagcattgcagatcgtctgaatgaactgatggttaaaccgcgtattgccaaagaatggaatcat aatagcgtgcgtgatatcctgaccaacgatatctatattggcacctatcgttggggtgataaagttgttccgaataatcat ccgcctattattagcgaaaccctgttcaaaaaagcccagaaagaaaaagaaaaacgtggcgttgatcgtaaacgcgttg gtaaatttctgtttaccggtctgctgcagtgtggtaattgtggtggccataaaatgcagggccattttgataaacgtgagc agaaaacctattaccgttgtaccaaatgtcaccgcattaccaacgaaaaaaacattctggaaccgctgctggatgaaatt cagctgctgattaccagcaaagaatactttatgagcaaattcagcgaccgctatgatcagcaagaggttgttgatgttag cgcactgacaaaagaactggaaaaaatcaaacgccagaaagagaaatggtacgatctgtatatggatgatcgtaaccc gattccgaaagaagaactgtttgccaaaattaacgaactgaacaaaaaagaagaagaaatctatagcaagctgagcga agtggaagaagataaagaaccggttgaagagaaatataaccgcctgagcaaaatgatcgattttaaacagcagtttga gcaggccaacgactttaccaaaaaagagctgctgttcagcatcttcgaaaagattgtgatttatcgcgagaaaggcaag ctgaaaaaaatcaccctggattacaccctgaaataa

Table F.2: Integrase site sequences.

Integrase	DNA sequence
Int3	atgcgtaaagtggcaatttatagccgtgtgagcaccattaatcaggcagaagaaggttatagcattcagggtcagattg aagcactgaccaaatattgtgaagccatggaatggaagatctataagaactatagtgatgccggttttagcggtggtaa actggaacgtccggcaattaccgaactgattgaagatggcaaaaacaacaaattcgataccatcctggtgtataaactg gatcgcctgagccgtaatgttaaagataccctgtatctggtgaaagatgtttttaccgccaacaacattcattttgtgagc ctgaaagaaaacatcgataccagcagcgcaatgggtaacctgtttctgacactgctgagcgcaattgcagaatttgaac gtgagcagattaaagaacgtatgcagtttggtgttatgaaccgtgcaaaaagcggtaaaaccaccgcatggaaaaccc ctccgtatggttatcgttataacaaagatgaaaaaaccctgagcgtgaatgaactggaagcagcaaatgttcgtcagat gtttgatatgattattagcggctgtagcatcatgagcattaccaattatgcacgcgataactttgttggtaatacctggac ccatgtgaaagtgaaacgtattctggaaaacgaaacctataaaggcctggtgaaatatcgtgaacagacctttagtggt gatcatcaggcaattattgacgaaaagacctacaacaaagcacagattgcactggcacatcgtaccgataccaaaacca atacccgtccgtttcagggcaaatatatgctgagccatattgccaaatgtggttattgtggtgcaccgctgaaagtttgta ccggtcgtgccaaaaatgatggcacccgtcgtcagacctatgtttgtgtgaataaaaccgaaagcctggcacgtcgtag cgtgaataattacaacaatcagaaaatctgcaacaccggtcgctatgagaaaaaacacatcgagaaatatgtgattgat gtgctgtacaaactgcagcacgataaagagtacctgaaaaaaatcaaaaaagatgataacattattgatattactccgc tgaaaaaagaaattgaaatcattgataaaaagattaatcgcctgaatgatctgtatattaacgacctgatcgatctgccg aaactgaaaaaggatatcgaagaactgaaccacctgaaagatgactacaacaaggccatcaaactgaactatctggac aaaaaaaacgaagatagcctgggtatgctgatggataatctggatattcgtaaaagcagctatgatgtgcagagccgta ttgtgaaacagctgattgatcgtgttgaagtgaccatggataatattgatatcatttttaagttctaa
Int4	atgattaccacccgtaaagtggcaatttatgtgcgtgttagcaccaccaatcaggcagaagaaggttatagcattcagg gtcagattgatagcctgatcaaatattgtgaagcaatgggctggatcatctatgaagaatataccgatgcaggttttagc ggtggtaaaattgatcgtccggcaatgagcaaactgattaccgatgccaaacataaacgctttgataccatcctggtgta taaactggatcgtctgagccgtagcgttcgtgataccctgtatctggttaaagatgtgttcaaccagaacaacatccattt tgttagcctgcaagaaaacattgataccagcagcgcaatgggtaacctgtttctgacactgctgagcgcaattgcagaat ttgaacgtgagcagattaccgaacgtatgaccatgggcaaaattggtcgtgcaaaaagcggtaaaaccatggcatgga cctataccccgtttggttatgattacaacaaagaaaaaggcgaactgattctggacccggcaaaagcaccgattgtgaa aatgatctataccgattatctgaaaggcatgagcatccagaaaatcgtggataaactgaataaaatggattataatggc aaagattgcacctggtttccgcatggtgttaaacatctgctggataatccggtgtattatggtatgacccgctataacaat aaactgtttccgggtaatcatcagccgatcattaccaaagaactgttcgataaaacccagcgtgaacgtcagcgtcgtcg tctgggtattgaagaaaatcattatacgattccgtttcaggccaaatacatgctgagcaaatttctgcgttgtcgtcagtg tggtagccgtatgggtctggaactgggtcgtccgcgtaaaaaagaaggtaaacgtagcaaaaaatactattgcctgaat agccgtccgaaacgtaccgcaagctgtgatacaccgctgtatgatgcagaaaccctggaagattatgtgctgcatgaaa ttgccaaaatccagaaagatccgagcattgcaagtcgccagaaacatattgaagatcacgagctgaaatacaaacgcg aacgtattgaagccaacatcaataaaaccgttaatcagctgtccaagctgaataatctgtatctgaatgatctgattacg ctggaagatctgaaaacccagaccaataccctgattgcaaaaaaacgcctgctggaaaatgaactggataaaacctgtg ataacgatgatgagctggatcgccaagaaaccattgcagactttctggcactgccggatgtttggaccatggattatgaa ggtcagaaatatgcagttgaactgctggttcagcgtgttaaagttgatcgcgataacatcgatatccactggacctttta a

Table F.2: Integrase site sequences.

Reporter	DNA sequence
BFP	ATGAGCGAGCTGATTAAGGAGAACATGCACATGAAGCTGTACATGGAGG GCACCGTGGACAACCATCACTTCAAGTGCACATCCGAGGGCGAAGGCAA GCCCTACGAGGGCACCCAGACCATGAGAATCAAGGTGGTCGAGGGCGGC CCTCTCCCCTTCGCCTTCGACATCCTGGCTACTAGCTTCCTCTACGGCAG CAAGACCTTCATCAACCACACCCAGGGCATCCCCGACTTCTTCAAGCAGT CCTTCCCTGAGGGCTTCACATGGGAGAGAGTCACCACATACGAAGACGG GGGCGTGCTGACCGCTACCCAGGACACCAGCCTCCAGGACGGCTGCCTC ATCTACAACGTCAAGATCAGAGGGGTGAACTTCACATCCAACGGCCCTG TGATGCAGAAGAAAACACTCGGCTGGGAGGCCTTCACCGAGACGCTGTA CCCCGCTGACGGCGGCCTGGAAGGCAGAAACGACATGGCCCTGAAGCTC GTGGGCGGGAGCCATCTGATCGCAAACATCAAGACCACATATAGATCCA AGAAACCCGCTAAGAACCTCAAGATGCCTGGCGTCTACTATGTGGACTA CAGACTGGAAAGAATCAAGGAGGCCAACAACGAGACCTACGTCGAGCAG CACGAGGTGGCAGTGGCCAGATACTGCGACCTCCCTAGCAAACTGGGGC ACTAA
mKate	ATGTCAGAATTAATTAAAGAAAATATGCACATGAAATTATATATGGAAG GTACTGTCAACAATCATCATTTCAAATGCACATCCGAAGGTGAAGGTAA ACCATATGAAGGCACACAAACAATGCGCATCAAAGCAGTTGAAGGTGGA CCCCTGCCCTTTGCGTTTGACATTCTCGCAACGAGCTTTATGTACGGGT CTAAAACTTTTATCAATCACACCCAAGGCATTCCTGACTTTTTTAAACAG TCCTTTCCTGAAGGCTTTACCTGGGAACGTGTAACAACTTATGAAGATG GCGGTGTACTTACAGCAACTCAAGATACGAGTTTACAAGATGGCTGTCT GATTTACAATGTTAAAATCCGTGGCGTAAATTTCCCGAGTAACGGACCC GTAATGCAAAAAAAAACTCTTGGTTGGGAAGCATCAACAGAAACCTTAT ATCCTGCGGACGGTGGCTTAGAAGGACGCGCAGACATGGCACTGAAATT AGTTGGAGGCGGTCATTTAATCTGCAACCTGAAAACAACCTATCGTTCC AAAAAACCCGCTAAAAACCTTAAAATGCCTGGAGTATACTATGTTGATC GTCGCTTAGAACGTATTAAAGAAGCTGATAAAGAAACCTACGTTGAACA ACATGAAGTAGCCGTAGCCCGTTATTGTGACCTTCCGTCGAAATTAGGA CATCGTTGATAA
sfGFPregistry	atgcgtaaaggcgaagagctgttcactggtgtcgtccctattctggtggaactggatggtgatgtcaacggtcataagtt ttccgtgcgtggcgagggtgaaggtgacgcaactaatggtaaactgacgctgaagttcatctgtactactggtaaactgc cggtaccttggccgactctggtaacgacgctgacttatggtgttcagtgctttgctcgttatccggaccatatgaagcagc atgacttcttcaagtccgccatgccggaaggctatgtgcaggaacgcacgatttcctttaaggatgacggcacgtacaaa acgcgtgcggaagtgaaatttgaaggcgataccctggtaaaccgcattgagctgaaaggcattgactttaaagaagacg gcaatatcctgggccataagctggaatacaattttaacagccacaatgtttacatcaccgccgataaacaaaaaaatgg cattaaagcgaattttaaaattcgccacaacgtggaggatggcagcgtgcagctggctgatcactaccagcaaaacact ccaatcggtgatggtcctgttctgctgccagacaatcactatctgagcacgcaaagcgttctgtctaaagatccgaacga gaaacgcgatcatatggttctgctggagttcgtaaccgcagcgggcatcacgcatggtatggatgaactgtacaaatga taa

Table F.3: Reporter sequences.

Short name	Original name	DNA sequence
T1	ECK120033737	ggaaacacagAAAAAAGCCCGCACCTGACAGTGCGGGCTTTTTTTTTcgaccaaagg
T2	ECK120029600	TTCAGCCAAAAAACTTAAGACCGCCGGTCTTGTCCACTACCTTGCAGTAATGCGGTGGACAGGATCGGCGGTTTTCTTTTCTCTTCTCAA
T3	L3S2P21	CTCGGTACCAAATTCCAGAAAAGAGGCCTCCCGAAAGGGGGGCCTTTTTTCGTTTTGGTCC
T4	L3S3P21	CCAATTATTGAAGGCCTCCCTAACGGGGGGCCTTTTTTTGTTTCTGGTCTCCC
T5	B0015	ccaggcatcaaataaaacgaaaggctcagtcgaaagactgggcctttcgttttatctgttgtttgtcggtgaacgctctctactagagtcacactggctcaccttcgggtgggcctttctgcgtttata
T6	J61048	ccggcttatcggtcagtttcacctgatttacgtaaaaacccgcttcggcgggttttgctttttggaggggcagaaagatgaatgactgtccacgacgctatacccaaaagaaa
T7	ECK120015170	ACAATTTTCGAAAAAACCCGCTTCGGCGGGTTTTTTTATAGCTAAAA
T8	ECK120010855	GTAACAACGGAAACCGGCCATTGCGCCGGTTTTTTTTGGCCT
T9	L3S2P11	CTCGGTACCAAATTCCAGAAAAGAGACGCTTTCGAGCGTCTTTTTTCGTTTTGGTCC
T10	L3S3P22	CCAATTATTGAAGGCCGCTAACGCGGCCTTTTTTTGTTTCTGGTCTCCC
	L3S1P13	gacgaacaataaggcctccctaacgggggggccttttttattgataacaaaa

Table F.5: List of terminator.

Name	DNA sequence
P7	AAAAAATTTATTTGCTTTCGCATCTTTTTGTACCTATAATGTGTGGA
P6	TTGACAATTAATCATCCGGCTCGTAATGTTTGTGGA
P5	ttgacaattaatcatccggctcgtaatttatgtgga
P2	aaaaagagtattgacttcgcatcttttgtacctataatgtgtgga
J23100	ttgacggctagctcagtcctaggtacagtgctagc
ProC	cacagctaacaccacgtcgtccctatctgctgccctaggtctatgagtggttgctggataactttacgggcatgcataaggctcgtatgatatattcagggagaccacaacggtttccctctacaaataattttgtttaacttt

Table F.6: List of promoters.

Name	DNA sequence
B0034	aaagaggagaaa
B0032	tcacacaggaaag
RBS-INT5	cagaggaaggaggctcg
RBS-INT7	agtaatttcaacaaaataactaggattcga
BCD2	GGGCCCAAGTTCACTTAAAAAGGAGATCAACAATGAAAGCAATTTTCGTACT GAAACATCTTAATCATGCTAAGGAGGTTTTCTA

Table F.7: List of RBSs.

Name	DNA sequence
RiboJ	AGCTGTCACCGGATGTGCTTTCCGGTCTGATGAGTCCGTGAGGACGAAACAG CCTCTACAAATAATTTTGTTTAA
BydvJ	AGGGTGTCTCAAGGTGCGTACCTTGACTGATGAGTCCGAAAGGACGAAACAC CCCTCTACAAATAATTTTGTTTAA
ElvJ	AGCCCCATAGGGTGGTGTGTACCACCCCTGATGAGTCCAAAAGGACGAAATG GGGCCTCTACAAATAATTTTGTTTAA
AraJ	AGTGGTCGTGATCTGAAACTCGATCACCTGATGAGCTCAAGGCAGAGCGAAA CCACCTCTACAAATAATTTTGTTTAA

Table F.8: List of ribozymes.

Name	DNA sequence
sp20-1	TAGTTGCGTCTCAGGGACCC
sp20-2	TAAGTGGCAATCCCGCCTGA
sp20-3	AAACCCGTCGCAGTATCCCT
sp20-4	ACTCAGGTCTGCCGTAAGGG
sp20-5	TGGAGGGCGAGGTTCCTTAC
sp20-6	AACCAGTGCTCTCGGTAGGG
sp20-7	CTCTGGCAGCCTGGTAGGTT
sp20-8	ATTGGGCTACAGTGTCCGCT
sp20-9	GAAGGACGGTGCGTTGTTCA
sp20-10	TTCCGTGTGCCAGAAAGTGC
sp20-11	AACAGTTCGTTGACCCGACG
sp20-12	AGATTGGTCCGAAGCAGGCT
sp20-13	AGGGATTTCGCCGTGACTCT
sp20-14	GAGTCTGACGAACGAGTGCG
sp20-15	AGACGGTCCCGCACCTTATT
sp20-16	CTTTCCGAGTGGAGGAGCCT
sp20-17	ATACGGACCCTCGTTGGCTT
sp20-18	AAGATTGAGCGTCCCGAGGT
sp20-19	CTGGGCAGAGCAGTTACCCT

Table F.9: List of 20bp spacers.

Name	DNA sequence
Spacer0	CTCGGATACCCTTACTCTGTTGAAAACGAATAGATAGGTT
Spacer1	TGCTCGTAGTTTACCACGGATACAGACAGTGATAATCTTA
Spacer2	AGATTACTACTGATAACCACTGTTGATTGGGATACCCGTA
Spacer3	AAGGAACGGTTATTTCTGCGTAGATCTATCTTACACAGCA
Spacer4	AGGCAACTGAAACGATTCGGATCCTGTATTACTATTCTTA
Spacer5	ACTTTATCTGAGAATAGTCAATCTTCGGAAATCCCAGGTG
Spacer5′	TAAAAGTCTCGTAAAGCGTTCTATCAATAACCCGTTGGTG
Spacer6	CCGTCTCAGAATCGGCCGTGAACAATAAAATAGTTTCGGT
Spacer7	TAATAAAAGGTCCCGTCTGAACTTACTGTGAATTCGACTA
Spacer8	GAATAATAGGAAGTTCGCCTGATTGTAAACACTCTCGTCT
Spacer9	CTGCTTTCCTTCTGATTGAGACGAGTAAAACACTGAATAG
Spacer10	TTGTAGCACTGTAAGATTTATCCACGAAGGTCAGCAACTT
Spacer11	AAAGTGCGGGTATTACAGTCTTATTTATCAGAACACCTGC
Spacer12	AAGGAACGGTTATTTCTGCGTAGACTTATCTTACACAGCA
Spacer13	AGGCAACTGAAACGATTCGGACGCTGTATTACTATTCTTA
Spacer14	TTTACCCGAATCTATTGAAACAGAGACGGAGTCGCTTTTA
Spacer15	CCGTCTCAGAATCTCGTGTGAACAATAAAATAGTTTCGGT
Spacer16	TAATAAAAGGTCCCGTCTGAACTTACTGTGATTGCGACTA
Spacer17	TGAAATACGAATCCGTTGAGTTCCCAGTGAAGTAATCTCT
Spacer18	GATACTGTTACTTACCGATTATTGTGAAGAACCAGACCGT
Spacer19	TCACTTTTATCGGTTTCCAGAACAGGTAAGAGCCAATAGT
Spacer20	CAGAAAGGTATTGTTTACAGGTGCGACGACTTCAACTATT
Spacer21	TTGTTGTAGCACACTCGGCGAAAATCTGAATAGTAACTTC
Spacer22	ATAAAGTTGTGCCGTATCCAGCGGTTACCAATAATAGTCT
Spacer23	TAGTAAAGTTCCAATAAGACTCCAGGTATCTGTCCGTGTA
Spacer24	ATTGAACCTCTACTACGAGTGAGTTGAGATTACAGCCTTA
Spacer25	CTTACGCTATTATTGAAGCCAGTCTGTTACCGAAGTGAAA
Spacer26	AACGATTACGGATTGCTCTACTGTGACTGAAGTTTACAAC
Spacer27	TGAGGCACAGAGATTTACTTTATTCACGACTTCAGATACG
Spacer28	AATACGGTCTACTACAGAAGGGTGGTTTATCTTACTCAAC
Spacer29	AGATTTCCCTCGTCACGCAGTAAGTATTTATCGTAGAAGA
Spacer30	GTTTCAACCAGAGGGATTACAACTCGTTTTACTCCGAATA
Spacer31	GCTTATTTCGTATTACAACGGTAGAATCAACTTCCAGAGG
Spacer32	TGAAAGGAATCTGGTCTTACTGTCTGAGTCACAATACGAT
Spacer33	GCGGTTCCTATCGTATTCGTCAGTTATCACAGAAGTAAAA
SpacerN	ATTATTGACCACTTCCGAGTAGAATCGTGCTTCAGTAAGA

Table F.10: List of 40bp spacers.

F.2 Primers

Number	DNA sequence
38	TAAGAATAGTAATACAGGATCCGAATCGTTTC
71	cgttttcaacagagtaagggtatccgag
72	attattgaccacttccgagtagaatcgtg
109	gattaactaataaggaggacaaacatgtc
224	atgtcagaactaatcaaagagaatatgcac
862	AACCTATCTATTCGTTTTCAACAGAGTAAGGGTATCCGAGtagcaatcaactcactggctc
863	attattgaccacttccgagtagaatcgtgcttcagtaagagtcactaagggttagttagttagattagc
870	caggtgtagtacctatcttggttg
871	GTTAACATCACCATCCAATTCAACC
1112	ggacgggagcaagacgtttc
1153	acggtctggttcttcacaataatcggtaagtaacagtatccagaaatcatccttagcgaaagctaag
1287	gatgtgcatttgaaatgatgattgttg
1288	ccagttccaccagaataggg
1289	atgcgtaaaggcgaagag
1290	gtcagaattaattaaagaaaatatgcacatg
1318	TTGACAATTAATCATCCGGCTCG
1319	TCCACAAACATTACGAGCCG
1322	ccagttccaccagaataggg
1323	gtaaaggcgaagagctgttcac
1324	ggctcgtaatttatgtggaaaagaggagaaatactaggTGAGAGCCCTGGTAGTCATCCG
1325	cggtgacgcgggacaggcggatgactaccagggctctcacctagtatttctcctcttttccacataaattac
1326	gcagcgtggtcgaacggctacacaccgggatgtcgtaaATaaagttgtgccgtatccagc
1327	gactattattggtaaccgctggatacggcacaactttatTTAcgacatcccggtgtgtag
1328	gtacctataatgtgtggatcacacaggaaagtactagATGactaagaaagtagcaatctatacac

Table F.11: List of Primers.

Number	DNA sequence
1329	ggatactcgtgtatagattgctactttcttagtcatCTagtactttcctgtgtgatccac
1330	gtagatatcatatttaaattccaactcgctaccggttaaATtgaacctctactacgagtgagttg
1331	taaggctgtaatctcaactcactcgtagtagaggttcaatttaACcggtagcgagttggaatttaaatatg
1332	ctcagtcctaggtacagtgctagccagaggaaggaggctcgATgcctggtatgaccaccg
1333	cctgcaggatccggaccggtttcggtggtcataccaggcatCGagcctccttcctctgg
1334	tgaacgcgttaaaatcaaatgggcagcccgtacctaaAACgattacggattgctctactg
1335	gttgtaaacttcagtcacagtagagcaatccgtaatcgttTTAggtacgggctgccca
1336	taactttagtaatttcaacaaaataactaggattcgaATGaaagtggccatttatgttcg
1337	gttcatcggtgctaacacgaacataaatggccactttcatTCgaatcctagttattttgttgaaattac
1338	ctgaaaaaaatcaccctggattacaccctgaaaTAAaatacggtctactacagaagggtg
1339	gagtaagataaaccacccttctgtagtagaccgtattTTAtttcagggtgtaatccaggg
1366	TAATAAAAGGTCCCGTCTGAACTTAC
1829	ggaaggtactgtcaacaatcatc
1832	cactccagccagctttcc
sp18F	gatactgttacttaccgattattgtg
sp18R	acggtctggttcttcacaataatc
sp19F	tcacttttatcggtttccagaacagg
sp19R	actattggctcttacctgttctggaaacc
sp23F	tagtaaagttccaataagactccagg
SP23R	tacacggacagatacctggag
sp25F	CTTACGCTATTATTGAAGCCAGTC
sp25R	TTTCACTTCGGTAACAGACTGG
sp27F	TGAGGCACAGAGATTTACTTTATTC
sp27R	cgtatctgaagtcgtgaataaagtaaatc

Table F.11: List of Primers.

F.3 DNA sequences of fragments

Name	DNA sequence
CM-N	GGTCAATCTGCCGCAATCCAGTCTGTATACCCTTACTCTGTTGAAAACGAAT AGATAGGTTGCTAGCGGATCCAGCTGTCACCGGATGTGCTTTCCGGTCTGAT GAGTCCGTGAGGACGAAACAGCCTCTACAAATAATTTTGTTTAAGGGCCCAA GTTCACTTAAAAAGGAGATCAACAATGAAAGCAATTTTCGTACTGAAACATC TTAATCATGCTAAGGAGGTTTTCTAatgtcaaaaggagaagaacttttacaggtgtagtacctatctt ggttgaattggatggtgatgttaacggtcacaaattttctgtaTACACTGGTTATCTCGGCACAGACGG
CM-P	GGTCAATCTGCCGCAATCCAGTCTGTATACCCTTACTCTGTTGAAAACGAAT AGATAGGTTGCTAGCAAAAAATTTATTTGCTTTCGCATCTTTTTGTACCTAT AATGTGTGGAGGATCCAGCTGTCACCGGATGTGCTTTCCGGTCTGATGAGTC CGTGAGGACGAAACAGCCTCTACAAATAATTTTGTTTAAGGGCCCAAGTTCA CTTAAAAAGGAGATCAACAATGAAAGCAATTTTCGTACTGAAACATCTTAAT CATGCTAAGGAGGTTTTCTAatgtcaaaaggagaagaacttttacaggtgtagtacctatcttggttgaat tggatggtgatgttaacggtcacaaattttctgtaTACACTGGTTATCTCGGCACAGACGG
L3S3P00	actttatctgagaatagtcaatcttcggaaatcccaggtggcatgctaaaagtctcgtaaagcgttctatcaataacccgttgg tgCCAATTATTGAAGGGGAGCGGGAAACCGCTCCCCTTTTTTTGTTTCTGGTC TCCCccgtctcagaatcggccgtgaacaataaaatagtttcggtattattgaccacttccgagtagaatcgtgcttcagtaa ga
SGb48	gtttgtgtcccttcataaggcttcccctcaccctcgcttgtacatttgaaatgatgattgtttaccgttccttccatgtacagcttc atgtgcatattctctttgattagttctgacattctaaaatcctccttaagagcTGATCAgagcgccggatcagggagtgga cggcctgggagcgttgacaacttgcgcaccctgatctgCGTGCGTCAATTTTGTCAAAATAATTTTA TTGACAACGTCTTATTAACGTTGATATAATTTAAATTTTATTTGACAAAAAT GGGCTCGTGTTGTACAATAAATGTCTCTAGTGgcaccgaccgcagccacagcgtgtagcgctcc caggagagttatcgacttgcgtattaggggctagcgattaactaataaggaggacaaacatgtcaaaaggagaagaacttttt acaggtgtagtacctatcttggttg
SGb49	attgtttaccgttccttccatgtacagcttcatgtgcatattctctttgattagttctgacattctaaaatcctccttaagagcTG ATCAagacgagaaacgttccgtccgtctgggtcagttgcctaaccttaacttttacgcaggttcagcttCGTGCGTC AATTTTGTCAAAATAATTTTATTGACAACGTCTTATTAACGTTGATATAATT TAAATTTTATTTGACAAAAATGGGCTCGTGTTGTACAATAAATGTCTCTAGT Gaacggacgacccggtcatcaactttgcccatgtaaacttaactctcacacaggtttataacacgctagcgattaactaataag gaggacaaacatgtcaaaaggagaagaacttttacaggtgtagtacctatcttggttg

Table F.12: List of fragments.

Name	DNA sequence
SGb68	gatactgttacttaccgattattgtgaagaaccagaccgtttgttgtagcacactcggcgaaaatctgaatagtaacttcttgac aattaatcatccggctcgtaatttatgtggaaaagaggagaaatactagataaagttgtgccgtatccagcggttaccaataat agtctccaggcatcaaataaaacgaaaggctcagtcgaaagactgggcctttcgttttatctgttgtttgtcggtgaacgctctc tactagagtcacactggctcaccttcgggtgggcctttctgcgtttatatagtaaagttccaataagactccaggtatctgtccgt gtaaaaaagagtattgacttcgcatcttttgtacctataatgtgtggatcacacaggaaagtactagattgaacctctactacg agtgagttgagattacagccttactcggtaccaaattccagaaaagagacgctttcgagcgtctttttcgttttggtcccttacg ctattattgaagccagtctgttaccgaagtgaaattgacggctagctcagtcctaggtacagtgctagccagaggaaggaggct cgaacgattacggattgctctactgtgactgaagtttacaacccaattattgaaggccgctaacgcggccttttttgtttctggt ctccctgaggcacagagatttactttattcacgacttcagatacgcacagctaacaccacgtcgtccctatctgctgccctaggtc tatgagtggttgctggataactttacgggcatgcataaggctcgtatgatatattcagggagaccacaacggtttccctctacaa ataattttgtttaactttagtaatttcaacaaaataactaggattcgaaatacggtctactacagaagggtggtttatcttactca acgacgaacaataaggcctccctaacgggggggccttttttattgataacaaaaagatttccctcgtcacgcagtaagtatttatc gtagaagatcacttttatcggtttccagaacaggtaagagccaatagt
SGb98	aggcaactgaaacgattcggatcctgtattactattcttatcacactggctcaccttcgggtgggcctttctgcgtttatatacta gagagagaatataaaaagccagattattaatccggctttttttattatttactttatctgagaatagtcaatcttcggaaatccca ggtggcgagttttttatttcgtttatttcaattaaggtaactaaaaaactcctttactcaggtctgccgtaaggggggtttgtaccg tacaccactgagaccgcggtggttgaccagacaaaccacgaattattgaccacttccgagtagaatcgtgcttcagtaaga
SGb99	ctcggatacccttactctgttgaaaacgaatagataggtttcggccggcttgtcgacgacggcggtctccgtcgtcaggatcatc cgggcttgacaattaatcatccggctcgtaatgtttgtggaatgccaacacaattaacatctcaatcaaggtaaatgctttttgc ttttttttgcaggcaactgaaacgattcggatcctgtattactattcttatcacactggctcaccttcgggtgggcctttctgcgttt atatactagagagagaatataaaaagccagattattaatccggctttttttattatttactttatctgagaatagtcaatcttcgga aatcccaggtgttatcaacgatgtcctaatttcgacggaaggtcacaataacgggctacggctacttcatgttgttcaacgtagg tttctttatcagcttctttaatacgttctaagcgacgatcaacatagtatactccaggcattttaaggtttttagcgggttttttgg aacgataggttgttttcaggttgcagattaaatgaccgcctccaactaatttcagtgccatgtctgcgcgtccttctaagccaccg tccgcaggatataaggtttctgttgatgcttcccaaccaagagtttttttttgcattacgggtccgttactcgggaaatttacgcc acggattttaacattgtaaatcagacagccatcttgtaaactcgtatcttgagttgctgtaagtacaccgccatcttcataagttg ttacacgttcccaggtaaagccttcaggaaaggactgtttaaaaaagtcaggaatgccttgggtgtgattgataaaagttttag acccgtacataaagctcgttgcgagaatgtcaaacgcaaagggcaggggtccaccttcaactgctttgatgcgcattgtttgtgt gccttcatatggtttaccttcaccttcggatgtgcatttgaaatgatgattgttgacagtaccttccatatataatttcatgtgcat attttctttaattaattctgacatctagtatttctcctcttttaaacaaaattatttgtagaggtggtttcgctctgccttgagctc atcaggtgatcgagtttcagatcacgaccacttcgtggtttgtctggtcaaccaccgcggtctcagtggtgtacggtacaaaccc actcaggtctgccgtaagggaaaggagttttttagttaccttaattgaaataaacgaaataaaaactcgcattattgaccacttc cgagtagaatcgtgcttcagtaaga

Table F.12: List of fragments.

Name	DNA sequence
SGb103	gacatcaccatccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatttctcctcttttt aaacaaaattatttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccctatgccaacaca attaacatctcaatcaaggtaaatgctttttgctttttttgcgggagaccagaaacaaaaaaaggccccccgttagggaggcctt caataattggtttctttgggtatagcgtcgtggacagtcattcatctttctgcccctccaaaagcaaaaacccgccgaagcggg tttttacgtaaatcaggtgaaactgaccgataagccgggagcgccggatcagggagtggacggcctgggagcgctacacgctg tggctgcggtcggtgctagttgcgtctcagggacccgcccggatgatcctgacgacggagaccgcggtggttgaccagacaaa ccacgaattattgaccacttccgagtagaatcgtgcttcagtaaga
SGb104	tagtcgaattcacagtaagttcagacgggacctttttattagggagaccagaaacaaaaaaaggccccccgttagggaggcctt caataattggtttctttgggtatagcgtcgtggacagtcattcatctttctgcccctccaaaagcaaaaacccgccgaagcggg tttttacgtaaatcaggtgaaactgaccgataagccggaccgaaactattttattgttcacggccgattctgagacgggagcgc cggatcagggagtggacggcctgggagcgctacacgctgtggctgcggtcggtgctagttgcgtctcagggacccgcccggat gatcctgacgacggagaccgccgtcgtcgacaagccggccgatccacaaacattacgagccggatgattaattgtcaa
SGb105	tagtcgaattcacagtaagttcagacgggacctttttattagggagaccagaaacaaaaaaaggccccccgttagggaggcctt caataattggtttctttgggtatagcgtcgtggacagtcattcatctttctgcccctccaaaagcaaaaacccgccgaagcggg tttttacgtaaatcaggtgaaactgaccgataagccggaccgaaactattttattgttcacggccgattctgagacgggagcgc cggatcagggagtggacggcctgggagcgctacacgctgtggctgcggtcggtgctagttgcgtctcagggacccgcccggat gatcctgacgacggagaccgcggtggttgaccagacaaaccacgaattattgaccacttccgagtagaatcgtgcttcagtaag a
SGb102	gacatcaccatccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatttctcctcttttt aaacaaaattatttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccctatgccaacaca attaacatctcaatcaaggtaaatgctttttgctttttttgcgggagaccagaaacaaaaaaaggccccccgttagggaggcctt caataattggtttctttgggtatagcgtcgtggacagtcattcatctttctgcccctccaaaagcaaaaacccgccgaagcggg tttttacgtaaatcaggtgaaactgaccgataagccgggagcgccggatcagggagtggacggcctgggagcgctacacgctg tggctgcggtcggtgctagttgcgtctcagggacccgcccggatgatcctgacgacggagaccgccgtcgtcgacaagccggcc gatccacaaacattacgagccggatgattaattgtcaa
SGb110	ccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatttctcctcttttaaacaaaatta tttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccctatgccaacacaattaacatctc aatcaaggtaaatgctttttgctttttttgcgggagaccagaaacaaaaaaaggccccccgttagggaggccttcaataattgg tttctttgggtatagcgtcgtggacagtcattcatctttctgcccctccaaaagcaaaaacccgccgaagcgggtttttacgtaa atcaggtgaaactgaccgataagccgggagcgccggatcagggagtggacggcctgggagcgttgacaacttgcgcaccctg atctgttgacaattaatcatccggctcgtaatgtttgtgga

Table F.12: List of fragments.

Name	DNA sequence
SGb111	ccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatttctcctcttttaaacaaaatta tttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccctatgccaacacaattaacatctc aatcaaggtaaatgctttttgctttttttgcgggagaccagaaacaaaaaaaggccccccgttagggaggccttcaataattgg tttcttttgggtatagcgtcgtggacagtcattcatctttctgcccctccaaaagcaaaaacccgccgaagcgggtttttacgtaa atcaggtgaaactgaccgataagccgggagcgccggatcagggagtggacggcctgggagcgttgacaacttgcgcaccctg atctgttgacaattaatcatccggctcgtaatgtttgtggatcggccggcttgtcgacgacggcggtctcagtggtgtacggtac aaaccccccttacggcagacctgagtaaaggagtttttagttaccttaattgaaataaacgaaataaaaactcgcagtggtcg tgatctgaaactcgatcacctgatgagctcaaggcagagcgaaaccacctctacaaataattttgtttaaaaagaggagaaat actagatgtcagaattaattaaagaaaatatgcacatgaaattatatatggaaggtactgtcaacaatcatcatttcaaatgca catccgaaggtgaagg
SGb112	ccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatttctcctcttttaaacaaaatta tttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccctatgccaacacaattaacatctc aattaaggtaactaaaaaactcctttactcaggtctgccgtaaggggggtttgtaccgtacaccactgagaccgccgtcgtcga caagccggccgatccacaaacattacgagccggatgattaattgtcaacagatcagggtgcgcaagttgtcaacgctcccagg ccgtccactccctgatccggcgctcccggcttatcggtcagtttcacctgatttacgtaaaaacccgcttcggcgggtttttgcttt tggaggggcagaaagatgaatgactgtccacgacgctatacccaaaagaaaccaattattgaaggcctccctaacgggggggcc ttttttgtttctggtctcccgcaaaaaaagcaaaaagcatttaccttgattgaaataaacgaaataaaaactcgcagtggtcgt gatctgaaactcgatcacctgatgagctcaaggcagagcgaaaccacctctacaaataatttgtttaaaaagaggagaaata ctagatgtcagaattaattaaagaaaatatgcacatgaaattatatatggaaggtactgtcaacaatcatcatttcaaatgcac atccgaaggtgaagg
SGb113	ccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatttctcctcttttaaacaaaatta tttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccctatgccaacacaattaacatctc aatcaaggtaaatgctttttgctttttttgcgggagaccagaaacaaaaaaaggccccccgttagggaggccttcaataattgg tttcttttgggtatagcgtcgtggacagtcattcatctttctgcccctccaaaagcaaaaacccgccgaagcgggtttttacgtaa atcaggtgaaactgaccgataagccggtcactccagccagctttccggcaccgcttctggtgccggaaaccaggcaaagcgcc
SGb73	ttgacaattaatcatccggctcgtaatgtttgtggatcggccggcttgtcgacgacggcggtctcagtggtgtacggtacaaacc ccccttacggcagacctgagtaaaggagtttttagttaccttaattgaaataaacgaaataaaaactcgcagtggtcgtgatct gaaactcgatcacctgatgagctcaaggcagagcgaaaccacctctacaaataatttgtttaaaaagaggagaaatactaga tgtcagaattaattaaagaaaatatgcacatgaaattatatatggaaggtactgtcaacaatcatcatttcaaatgcacatccg aaggtgaaggtaaaccatatgaaggcacacaaacaatgcgcatcaaagcagttgaaggtggacccctgccctttgcgtttgac attctcgcaacgagctttatgtacgggtctaaaacttttatcaatcacacccaaggcattcctgacttttttaaacagtcctttcct gaaggctttacctgggaacgtgtaacaacttatgaagatggcggtgtacttacagcaactcaagatacgagtttacaagatggc tgtctgatttacaatgttaaaatccgtggcgtaaatttcccgagtaacggacccgtaatgcaaaaaaaaactcttggttgggaa gcatcaacagaaaccttatatcctgcggacggtggcttagaaggacgcgcagacatggcactgaaattagttggaggcggtca tttaatctgcaacctgaaaacaacctatcgttccaaaaaacccgctaaaaaccttaaaatgcctggagtatactatgttgatcgt cgcttagaacgtattaaagaagctgataaagaaacctacgttgaacaacatgaagtagccgtagcccgttattgtgaccttccg tcgaaattaggacatcgttgataacacctgggatttccgaagattgactattctcagataaagtaaataataaaaaagccggat taataatctggctttttatattctctctctagtatataaacgcagaaaggcccacccgaaggtgagccagtgtgataagaatagt aatacaggatccgaatcgtttcagttgcct

Table F.12: List of fragments.

Name	DNA sequence
SGb74	taagaatagtaatacaggatccgaatcgtttcagttgcctttagtgccccagtttgctagggaggtcgcagtatctggccactgc cacctcgtgctgctcgacgtaggtctcgttgttggcctccttgattctttccagtctgtagtccacatagtagacgccaggcatctt gaggttcttagcgggtttcttggatctatatgtggtcttgatgtttgcgatcagatggctcccgcccacgagcttcagggccatgt cgtttctgccttccaggccgccgtcagcggggtacagcgtctcggtgaaggcctcccagccgagtgttttcttctgcatcacagg gccgttggatgtgaagttcacccctctgatcttgacgttgtagatgaggcagccgtcctggaggctggtgtcctgggtagcggtc agcacgcccccgtcttcgtatgtggtgactctctcccatgtgaagccctcagggaaggactgcttgaagaagtcggggatgccc tgggtgtggttgatgaaggtcttgctgccgtagaggaagctagtagccaggatgtcgaaggcgaaggggagagggccgccctc gaccaccttgattctcatggtctgggtgccctcgtagggcttgccttcgccctcggatgtgcacttgaagtgatggttgtccacgg tgccctccatgtacagcttcatgtgcatgttctccttaatcagctcgctcatctagtatttctcctctttttaaacaaaattatttgt agaggctgtttcgtcctcacggactcatcagaccggaaagcacatccggtgacagctgcccggatgatcctgacgacggagac cgcggtggttgaccagacaaaccacgaattattgaccacttccgagtagaatcgtgcttcagtaaga
SGb75	ctcggataccttactctgttgaaaacgaatagataggttttatcatttgtacagttcatccataccatgcgtgatgcccgctgcg gttacgaactccagcagaaccatatgatcgcgtttctcgttcggatctttagacagaacgctttgcgtgctcagatagtgattgt ctggcagcagaacaggaccatcaccgattggagtgttttgctggtagtgatcagccagctgcacgctgccatcctccacgttgtg gcgaattttaaaattcgctttaatgccatttttttgtttatcggcggtgatgtaaacattgtggctgttaaaattgtattccagctt atggcccaggatattgccgtcttctttaaagtcaatgcctttcagctcaatgcggtttaccagggtatcgccttcaaatttcactt ccgcacgcgttttgtacgtgccgtcatccttaaaggaaatcgtgcgttcctgcacatagccttccggcatggcggacttgaagaa gtcatgctgcttcatatggtccggataacgagcaaagcactgaacaccataagtcagcgtcgttaccagagtcggccaaggtac cggcagtttaccagtagtacagatgaacttcagcgtcagtttaccattagttgcgtcaccttcaccctcgccacgcacggaaaac ttatgaccgttgacatcaccatccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatt tctcctcttttaaacaaaattatttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccct atgccaacacaattaacatctcaattaaggtaactaaaaaactcctttactcaggtctgccgtaaggggggtttgtaccgtacac cactgagaccgccgtcgtcgacaagccggccgatccacaaacattacgagccggatgattaattgtcaa
SGb76	tccacaaacattacgagccggatgattaattgtcaagcaaaaaaagcaaaaagcatttaccttgattgaaataaacgaaataa aaactcgcagtggtcgtgatctgaaactcgatcacctgatgagctcaaggcagagcgaaaccacctctacaaataattttgttt aaaaagaggagaaatactagatgtcagaattaattaaagaaaatatgcacatgaaattatatatggaaggtactgtcaacaat catcatttcaaatgcacatccgaaggtgaaggtaaaccatatgaaggcacacaaacaatgcgcatcaaagcagttgaaggtgg acccctgccctttgcgtttgacattctcgcaacgagctttatgtacgggtctaaaacttttatcaatcacacccaaggcattcctg actttttaaacagtcctttcctgaaggctttacctgggaacgtgtaacaacttatgaagatggcggtgtacttacagcaactca agatacgagtttacaagatggctgtctgatttacaatgttaaaatccgtggcgtaaatttcccgagtaacggacccgtaatgca aaaaaaaactcttggttgggaagcatcaacagaaaccttatatcctgcggacggtggcttagaaggacgcgcagacatggcac tgaaattagttggaggcggtcatttaatctgcaacctgaaaacaacctatcgttccaaaaaacccgctaaaaaccttaaaatgc ctggagtatactatgttgatcgtcgcttagaacgtattaaagaagctgataaagaaacctacgttgaacaacatgaagtagccg tagcccgttattgtgaccttccgtcgaaattaggacatcgttgataacacctgggatttccgaagattgactattctcagataaa gtaaataataaaaaagccggattaataatctggctttttatattctctctctagtatataaacgcagaaaggcccacccgaaggt gagccagtgtgataagaatagtaatacaggatccgaatcgtttcagttgcct

Table F.12: List of fragments.

Name	DNA sequence
SGb77	ctcggataccttactctgttgaaaacgaatagataggttttatcatttgtacagttcatccataccatgcgtgatgcccgctgcg gttacgaactccagcagaaccatatgatcgcgtttctcgttcggatctttagacagaacgctttgcgtgctcagatagtgattgt ctggcagcagaacaggaccatcaccgattggagtgttttgctggtagtgatcagccagctgcacgctgccatcctccacgttgtg gcgaattttaaaattcgctttaatgccatttttttgtttatcggcggtgatgtaaacattgtggctgttaaaattgtattccagctt atggcccaggatattgccgtcttctttaaagtcaatgcctttcagctcaatgcggtttaccagggtatcgccttcaaatttcactt ccgcacgcgttttgtacgtgccgtcatccttaaaggaaatcgtgcgttcctgcacatagccttccggcatggcggacttgaagaa gtcatgctgcttcatatggtccggataacgagcaaagcactgaacaccataagtcagcgtcgttaccagagtcggccaaggtac cggcagtttaccagtagtacagatgaacttcagcgtcagtttaccattagttgcgtcaccttcaccctcgccacgcacggaaaac ttatgaccgttgacatcaccatccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatt tctcctctttttaaacaaaattatttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccct atgccaacacaattaacatctcaattaaggtaactaaaaaactcctttactcaggtctgccgtaaggggggtttgtaccgtacac cactgagaccgcggtggttgaccagacaaaccacgaattattgaccacttccgagtagaatcgtgcttcagtaaga
SGb78	ctcggataccttactctgttgaaaacgaatagataggttttatcatttgtacagttcatccataccatgcgtgatgcccgctgcg gttacgaactccagcagaaccatatgatcgcgtttctcgttcggatctttagacagaacgctttgcgtgctcagatagtgattgt ctggcagcagaacaggaccatcaccgattggagtgttttgctggtagtgatcagccagctgcacgctgccatcctccacgttgtg gcgaattttaaaattcgctttaatgccatttttttgtttatcggcggtgatgtaaacattgtggctgttaaaattgtattccagctt atggcccaggatattgccgtcttctttaaagtcaatgcctttcagctcaatgcggtttaccagggtatcgccttcaaatttcactt ccgcacgcgttttgtacgtgccgtcatccttaaaggaaatcgtgcgttcctgcacatagccttccggcatggcggacttgaagaa gtcatgctgcttcatatggtccggataacgagcaaagcactgaacaccataagtcagcgtcgttaccagagtcggccaaggtac cggcagtttaccagtagtacagatgaacttcagcgtcagtttaccattagttgcgtcaccttcaccctcgccacgcacggaaaac ttatgaccgttgacatcaccatccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatt tctcctctttttaaacaaaattatttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccct atgccaacacaattaacatctcaatcaaggtaaatgctttttgctttttttgctagtcgaattcacagtaagttcagacgggacct tttattaggaccaaaacgaaaaaaggccccccctttcgggaggcctcttttctggaatttggtaccgagaccgaaactattttatt gttcacggccgattctgagacgg
SGb79	accgaaactattttattgttcacggccgattctgagacggtcactccagccagctttccggcaccgcttctggtgccggaaacca ggcaaagcgccattcgccattcaggctgcgcaactgttgggaagggcgatcggtgcgggcctcttcgctattacgccagctggc gaaaaggggatgtgctgcaaggcgattaagttgggtaacgccagggttttcccagtcacgacgttgtaaaacgacggccagtg aatccgtaatcatggtcatctagtatttctcctcttttaaacaaaattatttgtagaggccccatttcgtccttttggactcatca ggggtggtacacaccaccctatggggctgagcgccggatcagggagtggacggcctgggagcgctacacgctgtggctgcggt cggtgctagttgcgtctcagggacccgcccggatgatcctgacgacggagaccgccgtcgtcgacaagccggccgatccacaa acattacgagccggatgattaattgtcaa

Table F.12: List of fragments.

Name	DNA sequence
SGb80	tccacaaacattacgagccggatgattaattgtcaacagatcagggtgcgcaagttgtcaacgctcccaggagagttatcgact tgcgtattagggagctgtcaccggatgtgctttccggtctgatgagtccgtgaggacgaaacagcctctacaaataattttgttt aaaaagaggagaaatactagatgagcgagctgattaaggagaacatgcacatgaagctgtacatggagggcaccgtggaca accatcacttcaagtgcacatccgagggcgaaggcaagccctacgagggcacccagaccatgagaatcaaggtggtcgaggg cggccctctccccttcgccttcgacatcctggctactagcttcctctacggcagcaagaccttcatcaaccacacccagggcatcc ccgacttcttcaagcagtccttccctgagggcttcacatgggagagagtcaccacatacgaagacgggggcgtgctgaccgcta cccaggacaccagcctccaggacggctgcctcatctacaacgtcaagatcagaggggtgaacttcacatccaacggccctgtga tgcagaagaaaacactcggctgggaggccttcaccgagacgctgtaccccgctgacggcggcctggaaggcagaaacgacat ggccctgaagctcgtgggcgggagccatctgatcgcaaacatcaagaccacatatagatccaagaaacccgctaagaacctca agatgcctggcgtctactatgtggactacagactggaaagaatcaaggaggccaacaacgagacctacgtcgagcagcacga ggtggcagtggccagatactgcgacctccctagcaaactggggcactaaaggcaactgaaacgattcggatcctgtattactat tctta
SGb81	accgaaactattttattgttcacggccgattctgagacggtcactccagccagctttccggcaccgcttctggtgccggaaacca ggcaaagcgccattcgccattcaggctgcgcaactgttgggaagggcgatcggtgcgggcctcttcgctattacgccagctggc gaaaggggatgtgctgcaaggcgattaagttgggtaacgccagggttttcccagtcacgacgttgtaaaacgacggccagtg aatccgtaatcatggtcatctagtatttctcctcttttaaacaaaattatttgtagaggccccatttcgtccttttggactcatca ggggtggtacacaccaccctatggggctgagcgccggatcagggagtggacggcctgggagcgttgacaacttgcgcaccctg atctgttgacaattaatcatccggctcgtaatgtttgtgga
SGb82	ttgacaattaatcatccggctcgtaatgtttgtggatcggccggcttgtcgacgacggcggtctccgtcgtcaggatcatccggg cgggtccctgagacgcaactagcaccgaccgcagccacagcgtgtagcgctcccaggagagttatcgacttgcgtattaggga gctgtcaccggatgtgctttccggtctgatgagtccgtgaggacgaaacagcctctacaaataattttgtttaaaaagaggaga aatactagatgagcgagctgattaaggagaacatgcacatgaagctgtacatggagggcaccgtggacaaccatcacttcaag tgcacatccgagggcgaaggcaagccctacgagggcacccagaccatgagaatcaaggtggtcgagggcggccctctcccctt cgccttcgacatcctggctactagcttcctctacggcagcaagaccttcatcaaccacacccagggcatccccgacttcttcaag cagtccttccctgagggcttcacatgggagagagtcaccacatacgaagacgggggcgtgctgaccgctacccaggacaccag cctccaggacggctgcctcatctacaacgtcaagatcagaggggtgaacttcacatccaacggccctgtgatgcagaagaaaa cactcggctgggaggccttcaccgagacgctgtaccccgctgacggcggcctggaaggcagaaacgacatggccctgaagctc gtgggcgggagccatctgatcgcaaacatcaagaccacatatagatccaagaaacccgctaagaacctcaagatgcctggcgt ctactatgtggactacagactggaaagaatcaaggaggccaacaacgagacctacgtcgagcagcacgaggtggcagtggcc agatactgcgacctccctagcaaactggggcactaaaggcaactgaaacgattcggatcctgtattactattctta

Table F.12: List of fragments.

Name	DNA sequence
SGb83	taagaatagtaatacaggatccgaatcgtttcagttgcctttagtgccccagtttgctagggaggtcgcagtatctggccactgc cacctcgtgctgctcgacgtaggtctcgttgttggcctccttgattctttccagtctgtagtccacatagtagacgccaggcatctt gaggttcttagcgggtttcttggatctatatgtggtcttgatgtttgcgatcagatggctcccgcccacgagcttcagggccatgt cgtttctgccttccaggccgccgtcagcggggtacagcgtctcggtgaaggcctcccagccgagtgttttcttctgcatcacagg gccgttggatgtgaagttcacccctctgatcttgacgttgtagatgaggcagccgtcctggaggctggtgtcctgggtagcggtc agcacgccccgtcttcgtatgtggtgactctctcccatgtgaagccctcagggaaggactgcttgaagaagtcggggatgccc tgggtgtggttgatgaaggtcttgctgccgtagaggaagctagtagccaggatgtcgaaggcgaaggggagagggccgccctc gaccaccttgattctcatggtctgggtgccctcgtagggcttgccttcgccctcggatgtgcacttgaagtgatggttgtccacgg tgccctccatgtacagcttcatgtgcatgttctccttaatcagctcgctcatctagtatttctcctctttttaaacaaaattatttgt agaggctgtttcgtcctcacggactcatcagaccggaaagcacatccggtgacagctccctaatacgcaagtcgataactctcc tgggagcgctacacgctgtggctgcggtcggtgctagttgcgtctcagggacccgcccggatgatcctgacgacggagaccgcg gtggttgaccagacaaaccacgaattattgaccacttccgagtagaatcgtgcttcagtaaga
SGb84	ccgtctcagaatcggccgtgaacaataaaatagtttcggtctcggtaccaaattccagaaaagaggcctcccgaaaggggggc ctttttcgttttggtcctaataaaaggtcccgtctgaacttactgtgaattcgactagcaaaaaaagcaaaaagcatttaccttg attgaaataaacgaaataaaaactcgcagtggtcgtgatctgaaactcgatcacctgatgagctcaaggcagagcgaaaccac ctctacaaataattttgtttaaaaagaggagaaatactagatgtcagaattaattaaagaaaatatgcacatgaaattatatat ggaaggtactgtcaacaatcatcatttcaaatgcacatccgaaggtgaaggtaaaccatatgaaggcacacaaacaatgcgca tcaaagcagttgaaggtggacccctgccctttgcgtttgacattctcgcaacgagctttatgtacgggtctaaaacttttatcaat cacacccaaggcattcctgacttttttaaacagtcctttcctgaaggctttacctgggaacgtgtaacaacttatgaagatggcg gtgtacttacagcaactcaagatacgagtttacaagatggctgtctgatttacaatgttaaaatccgtggcgtaaatttcccgag taacggacccgtaatgcaaaaaaaaactcttggttgggaagcatcaacagaaaccttatatcctgcggacggtggcttagaag gacgcgcagacatggcactgaaattagttggaggcggtcatttaatctgcaacctgaaaacaacctatcgttccaaaaaacccg ctaaaaaccttaaaatgcctggagtatactatgttgatcgtcgcttagaacgtattaaagaagctgataaagaaacctacgttg aacaacatgaagtagccgtagcccgttattgtgaccttccgtcgaaattaggacatcgttgataacacctgggatttccgaaga ttgactattctcagataaagtaaataataaaaaagccggattaataatctggctttttatattctctctctagtatataaacgcag aaaggcccacccgaaggtgagccagtgtgataagaatagtaatacaggatccgaatcgtttcagttgcct
SGb85	accgaaactattttattgttcacggccgattctgagacggtcactccagccagctttccggcaccgcttctggtgccggaaacca ggcaaagcgccattcgccattcaggctgcgcaactgttgggaagggcgatcggtgcgggcctcttcgctattacgccagctggc gaaaggggatgtgctgcaaggcgattaagttgggtaacgccagggttttcccagtcacgacgttgtaaaacgacggccagtg aatccgtaatcatggtcatctagtatttctcctcttttaaacaaaattatttgtagaggccccatttcgtccttttggactcatca ggggtggtacacaccaccctatggggctgagcgccggatcagggagtggacggcctgggagcgctacacgctgtggctgcggt cggtgctagttgcgtctcagggacccgcccggatgatcctgacgacggagaccgcggtggttgaccagacaaaccacgaatta ttgaccacttccgagtagaatcgtgcttcagtaaga

Table F.12: List of fragments.

Name	DNA sequence
SGb52	ctcggataccttactctgttgaaaacgaatagataggttttatcatttgtacagttcatccataccatgcgtgatgcccgctgcg gttacgaactccagcagaaccatatgatcgcgtttctcgttcggatctttagacagaacgctttgcgtgctcagatagtgattgt ctggcagcagaacaggaccatcaccgattggagtgttttgctggtagtgatcagccagctgcacgctgccatcctccacgttgtg gcgaattttaaaattcgctttaatgccatttttttgtttatcggcggtgatgtaaacattgtggctgttaaaattgtattccagctt atggcccaggatattgccgtcttctttaaagtcaatgcctttcagctcaatgcggtttaccagggtatcgccttcaaatttcactt ccgcacgcgttttgtacgtgccgtcatccttaaaggaaatcgtgcgttcctgcacatagccttccggcatggcggacttgaagaa gtcatgctgcttcatatggtccggataacgagcaaagcactgaacaccataagtcagcgtcgttaccagagtcggccaaggtac cggcagtttaccagtagtacagatgaacttcagcgtcagtttaccattagttgcgtcaccttcaccctcgccacgcacggaaaac ttatgaccgttgacatcaccatccagttccaccagaatagggacgacaccagtgaacagctcttcgcctttacgcatctagtatt tctcctcttttaaacaaaattatttgtagaggggtgtttcgtcctttcggactcatcagtcaaggtacgcaccttgagacaccct atgccaacacaattaacatctcaatcaaggtaaatgctttttgctttttttgcttgacaattaatcatccggctcgtaatgtttgtg ga
SGb53	ttgacaattaatcatccggctcgtaatgtttgtggatcggccggcttgtcgacgacggcggtctccgtcgtcaggatcatccgggg cagctgtcaccggatgtgctttccggtctgatgagtccgtgaggacgaaacagcctctacaaataatttttgtttaaaaagaggga gaaatactagatgagcgagctgattaaggagaacatgcacatgaagctgtacatggagggcaccgtggacaaccatcacttca agtgcacatccgagggcgaaggcaagccctacgagggcacccagaccatgagaatcaaggtggtcgagggcggccctctccc cttcgccttcgacatcctggctactagcttcctctacggcagcaagaccttcatcaaccacacccagggcatccccgacttcttca agcagtccttccctgagggcttcacatgggagagagtcaccacatacgaagacgggggcgtgctgaccgctacccaggacacc agcctccaggacggctgcctcatctacaacgtcaagatcagaggggtgaacttcacatccaacggccctgtgatgcagaagaa aacactcggctgggaggccttcaccgagacgctgtaccccgctgacggcggcctggaaggcagaaacgacatggccctgaagc tcgtgggcgggagccatctgatcgcaaacatcaagaccacatatagatccaagaaacccgctaagaacctcaagatgcctggc gtctactatgtggactacagactggaaagaatcaaggaggccaacaacgagacctacgtcgagcagcacgaggtggcagtgg ccagatactgcgacctccctagcaaactggggcactaaaggcaactgaaacgattcggatcctgtattactattctta
SGb54	aggcaactgaaacgattcggatcctgtattactattcttatcacactggctcaccttcgggtgggcctttctgcgtttatatacta gagagagaatataaaaagccagattattaatccggctttttattatttactttatctgagaatagtcaatcttcggaaatccca ggtgttatcaacgatgtcctaatttcgacggaaggtcacaataacgggctacggctacttcatgttgttcaacgtaggtttcttta tcagcttctttaatacgttctaagcgacgatcaacatagtatactccaggcattttaaggtttttagcgggttttttggaacgata ggttgttttcaggttgcagattaaatgaccgcctccaactaatttcagtgccatgtctgcgcgtccttctaagccaccgtccgcag gatataaggtttctgttgatgcttcccaaccaagagtttttttttgcattacgggtccgttactcgggaaatttacgccacggatt ttaacattgtaaatcagacagccatcttgtaaactcgtatcttgagttgctgtaagtacaccgccatcttcataagttgttacacg ttcccaggtaaagccttcaggaaaggactgtttaaaaaagtcaggaatgccttgggtgtgattgataaaagttttagacccgta cataaagctcgttgcgagaatgtcaaacgcaaagggcaggggtccaccttcaactgctttgatgcgcattgtttgtgtgccttca tatggtttaccttcaccttcggatgtgcatttgaaatgatgattgttgacagtaccttccatatataatttcatgtgcatattttctt taattaattctgacatctagtatttctcctcttttaaacaaaattatttgtagaggtggtttcgctctgccttgagctcatcaggt gatcgagtttcagatcacgaccactgcgagttttatttcgtttatttcaattaaggtaactaaaaaactcctttactcaggtctg ccgtaaggggggtttgtaccgtacaccactgagaccgcggtggttgaccagacaaaccacgaattattgaccacttccgagtag aatcgtgcttcagtaaga

Table F.12: List of fragments.

Annex G

Protocols

Name: Pauline Mayonove
Date: 02/08/2017
Last validation/update: 26/06/2018 Sarah Guiziou
Team Synthetic Biology protocols

Gibson Assembly Mix preparation

Materials

- 1,5mL Microtubes
- PCR tubes

To prepare the 5X Isothermal solution (ISO 5X):

- Tris-HCl pH 7.5 solution 1M (on bench)
- PEG-8000 (Common powders in the JB's lab)
- $MgCl_2$ solution 1M (on bench)
- DTT 1M (common -20°C in GA Mix preparation Box)
- dNTP Mix 10mM (common -20°C in GA Mix preparation Box)
- NAD 100mM (aliquots in common -20°C in GA Mix preparation Box or powder of B-Nicotinamide adenine dinucleotide sodium sulfate with common powders in the JB's lab)
- dd H2O

To prepare the Gibson Assembly Mix 2X solution:

- ISO 5x (common -20°C in GA Mix preparation Box)
- T5 exonucelase (10 U/µL) (common -20°C in GA Mix preparation Box)
- Taq DNA ligase (40 U/µL) (common -20°C in GA Mix preparation Box)
- Phusion DNA polymerase (2U/µL) (common -20°C in GA Mix preparation Box)
- dd H2O

Protocol

5X isothermal reaction buffer.

Preparation for 10mL final solution

	10mL final	Final Conc
Tris-HCl pH 7.5	5mL	500mM
PEG-8000	2,5g	50%
Vortex		
MgCl2	500µL	50mM
DTT	500µL	50mM
dNTP Mix	1mL	1mM
NAD	500µL	5mM
ddH2O	Qsp 10mL	

The 5X isothermal reaction buffer has to be aliquot in 1mL, in 1,5mL tubes.

2X Gibson Assembly mix.

	x2	x2	
Final volume (µL)	1000	1500	
ISO 5x (µL)	300.8	451.2	
T5 exonucelase (10 U/µL)	0,6	0,9	Or do a dilution 1/10 of the solution 10U/µL and add 9µL for 1500µL final
Taq DNA ligase (40 U/µL)	150,4	225,6	
Phusion DNA polymerase (2U/µL)	18,8	28,2	
dd H2O	529.4	794.1	

The Gibson Assembly 2X mix has to be aliquot in 10µl in PCR tubes on a rack.
Put the rack at -20°C. When it's freeze put all tubes in a "tips" box.

Name: Pauline Mayonove
Date: 21/08/2017
Last validation/update: 26/06/2018 Sarah Guiziou

Cloning by Gibson Assembly

Steps

1. Design
2. PCR Q5
3. DpnI Digestion
4. PCR Clean up
5. Gibson Assembly
6. Transformation
7. Verifications
8. Storage

Principle

Gibson Assembly is allows for successful assembly of multiple DNA fragments, regardless of fragment length or end compatibility. It is ease-of-use, flexible and suitable for large DNA constructs.

Following the steps for cloning:
1 - **Design**

2 – **PCR Q5** amplification with primers containing the restriction sites, Agar Gel verification

3 – **DpnI Digestion** of the PCR to digest the template PCR DNA, as DpnI digest phosphorylated DNA.

4 – DNA used for Gibson Assembly has to be clean of enzymes and salts. **Clean up PCR** has to be performed.

5 – **Gibson Assembly** efficiently joins multiple overlapping DNA fragments in a single-tube isothermal reaction. The Gibson Assembly Master Mix includes three different enzymatic activities that perform in a single buffer:

- The exonuclease creates single-stranded 3´ overhangs that facilitate the annealing of fragments that share complementarity at one end (overlap region).
- The proprietary DNA polymerase fills in gaps within each annealed fragment.
- The DNA ligase seals nicks in the assembled DNA.

The end result is a double-stranded fully sealed DNA molecule.

6 – Bacterial **transformation** is perform to put the cloned DNA inside bacterial cells that will amplify it. You can transform using Electro or chemical-competent cells depend on the number of colonies required.
You should use chemical competent cells and if you don't have a good efficiency, you can try with electro-competent cells.

7 – Verifications by **Colony PCR** to find cells containing your cloned DNA and by **plasmid extraction, sequencing.**

Materials and Protocols

2 - PCR Q5

Materials

- Q5® Hot Start High-Fidelity DNA polymerase (NEB)
- Primers 20µM
- Template adjust to 1ng/µL
- Ultrapure Water
- PCR tubes
- PCR machine
- Agar gel material

Protocol

Depending on the final quantity of fragment needed and PCR yield, you can perform either a PCR with 20µL, 40µL or several times 40µL.

Mix 8µL Water + 10µL Q5 + 1µL Template + 0.5µL each Primer.

STEP	TEMP	TIME
Initial Denaturation	98°C	30 seconds
	98°C	10 seconds
30 Cycles	*50–72°C	20 seconds Temperature depend of primers
	72°C	30 seconds/kb Time according to the fragment size
Final Extension	72°C	2 minutes
Hold	12°C	

*Use of the NEB T_m Calculator is highly recommended.

. Prepare a 0.8% agars gel for sample larger than 1kb and 1.5% for sample smaller than 1kb.
. Load 2.5µL of the PCR reaction with 0.5µL of Loading dye in the gel and a 1kb ladder or 100bp ladder according to the size of your expected fragment.
. Image the gel.

3 - DpnI Digestion

Materials: DpnI (NEB)

Protocol

. Add 1µL DpnI in 17.5µL Tube from Q5.
. Mix well by pipetting up and down so that you solution is homogenous and glycerol is not in the bottom of the tube.
. 1h at 37°C
. 10min at 80°C and Hold at 12°C

4 - PCR Clean-up

Materials

- Biosentec PCR Clean-up Kit

Protocol

. Read the kit protocol (Keep resuspension buffer at 4°C, elute in 25µL EB at 37°C, place EB in the center of column and wait 5min before elution).
. Measure the DNA concentration with Nanodrop.

5 – Gibson Assembly

Materials

- 2X Gibson Assembly Mix (see Gibson Assembly Mix protocol)
- Insert fragment(s) from PCR or ordered.
- Vector fragment (after PCR, DpnI and clean-up).
- Ultrapure water
- Thermo-cycler at 50°C

Protocol

. Calculate the volume of insert and vector to mix in the Gibson Assembly reaction. For 2 fragment assembly (one insert and one vector), the optimum is to mix 100ng of vector with 3 times more insert in mole, and for more than 2 fragment assembly, 100ng of vector with same quantity of each insert in mole. The total volume of insert(s) and vector have to be 10µL or less, if the calculated total volume is higher, the quantity of vector can be reduced up to 50ng and the volume of insert calculated accordingly.
. Mix the vector and insert(s) fragments according to previously calculated proportions with 10µL of the Gibson Assembly Mix and adjust the total volume of the reaction to 20µL with water. As negative control of assembly, mix the vector alone in the same proportion than previously with 10µL of Gibson Assembly Mix and adjust the total volume of the reaction to 20µL with water.
. Place the reactions at 50°C during one hour.

6 – Transformation

Materials

- Chemical competent cells
- Water bath at 42°C
- Ice
- SOC
- Petri dish with LB agar medium and the appropriate antibiotic
- centrifuge

Protocol

1. Thaw gently competent cells on ice (aliquot of 100µL), one tube per Gibson assembly reaction (do not forget negative controls).
2. Add 10µL of the Gibson Assembly reaction (keep cells on ice)
3. Incubate 30min on ice
4. Heat-shock cells at 42°C during 45s (in water bath)
5. Put back on ice after heat-shock (2 to 5min)
6. Add 900 µL pre-warmed SOC (37°C)(rich medium)
7. Incubate cells at 37°C with agitation during at least 30min
8. Centrifuge cells at 4000rpm during 1min, remove 800µL of supernatant and plate the rest.
9. Incubate at 37°C overnight

Note

For multiple transformation (more than 10), competent cells aliquoted in PCR strip can be used. A similar protocol is used. To adapt to large volumes, cells are incubated in SOC in 96 well

plates and centrifugation is therefore performed in a centrifuge adapted for plates. Cells are plated in 6 well plates filled with 3mL of LB agar supplemented with appropriate antibiotics.

WARMING:
After transformation, count the number of colonies for the negative control plate (negative control of the Gibson assembly) and for the cloning. The ratio of the number of colonies for the cloning over the negative control should be higher than 10.

- If no colonies are obtained for both, the transformation can be re-performed using electro-competent cells, to increase the transformation efficiency.
- If the ratio between colonies in the negative control and for the cloning is lower than 5, the protocol should be stop here and previous steps should be debug.
- If the ratio is between 5 and 10, more colonies should be picked for colony PCR.

7 – Colony PCR

Materials

- 2X One-taq quick load master mix (NEB)
- Primers at 20µM
- Colonies
- PCR tubes
- PCR machine
- Agar gel material

Protocol

. For each cloning, perform two colony PCR from two different colonies.
. For each colony PCR, pick one colony and re-suspend it in 10µL of sterile water (in PCR tube).
. Pre-mix the One-Taq master mix, primers and water for the corresponding number of reaction, such as for one reaction: 5uL of master mix, 0.25µL of each primer, 3.5µL of water.
. Keep the colony re-suspended in water at 4°C, to use afterward to inoculate the culture for plasmid extraction.
. Mix 9uL of the pre-mix with 1L of the re-suspended colony.
. Place the tube in the PCR machine with the following PCR cycle:

- 95°C 5min
- 95°C 20 sec
- Temperature dependent on primers – 30sec
- 68°C – 1min/kb

Cycle 30 Times the 3 last steps.

- 68°C 5min
- Hold at 12°C

. Prepare a 0.8% agars gel for sample larger than 1kb and 1.5% for sample smaller than 1kb.
. Load directly 5µL of the PCR reaction in the gel and a 1kb ladder or 100bp ladder according to the size of your expected fragment.
. Image the gel.

8 – Plasmid extraction

Materials

- LB with appropriate antibiotic
- Falcon tubes
- Incubator
- Plasmid extraction kit from Qiagen

Protocol
. For the colony PCR with the corresponding fragment size, mix 5µL of the re-suspended colony in water in 5mL of LB for high copy plasmid or 10mL for low copy plasmid in a 50mL falcon.
. Place the culture at 37°C with agitation overnight.
. From the overnight culture, perform a strick of each culture in a petri dish with the appropriate antibiotic for further glycerol stock (Plate at 37°C overnight, and stored at 4°C)
. Centrifuge the culture (5min at 5000rpm) and perform plasmid extraction according to protocol from Qiagen kit.

Note
For low copy plasmid, double re-suspension, lysis and neutralization volume and elute in 30µL.

9 – Sequencing and glycerol stock

Materials
- GATC barcodes
- Tubes
- 50% glycerol

Protocol
. Send the extracted plasmid DNA to sequencing with the appropriate primers to verify the full cloned sequence, follow GATC procedure.
. For correct sequence, inoculate 2mL of LB with appropriate antibiotic with a colony from the corresponding strick.
. Place the culture at 37°C with agitation during 6 hours, until the culture is trouble.
. Mix 1.2mL of culture with 400µL of 50% glycerol (15% glycerol).
. Annotate the tube.
. Place the glycerol stock at -80°C, and register the corresponding information in the excel file of the glycerol stock box.

Name: Pauline Mayonove
Date: 08/01/2018
Last validation/update: 23/05/2018 Sarah Guiziou
Team Synthetic Biology protocols

Chemical competent cells preparation and Transformation with chemical competent cells E.coli

Materials

- LB medium
- TSS medium: To make 50 mL: 5g PEG 8000, 1.5 mL 1M $MgCl_2$ (or 0.30g $MgCl_2$*$6H_20$), 2.5 mL DMSO and LB to 50 mL. Filter sterilized (0.22 µm filter).
- LB plates
- 1.5mL tubes
- 50mL falcons
- Ice
- 500mL flask for culture
- Liquid nitrogen

Protocol

Day -1

- Steak an LB plate (with ATB if needed) with the E. coli strain

Day 0

- Inoculate 5mL of LB (with ATB if needed) with the strain from the fresh steak plate
- Incubated overnight at 37°C

Day 1

- Dilute the overnight culture into 50mL of LB without antibiotic at 1/500 (200µL in 100mL)
- Incubated at 37°C until OD650nm reach 0.2-0.3 (3-4h)
- Place 1.5mL tubes, racks, 10mL Pipettes at -20°C
- **DO EVERYTHING ON ICE**
- Incubate the culture on ice for 10min in 50mL falcon tubes
- Cold down the centrifuge to 4°C
- Centrifuge the culture at 3000rpm 4°C for 10min
- Remove the supernatant
- Resuspend cells in 10% volume of TSS buffer
- Aliquot cells in 100uL in 1.5mL tubes or in PCR tubes with multi-distribution pipette
- Freeze them with liquid nitrogen
- Store at -80°C

To test them: use pUC19 as positive control and do not forget negative control.

- Thaw gently 3 tubes of cells on ice
- Add 1µL PUC19 in 1 tube, add nothing in the others (negative control) (keep cells on ice)
- Incubate 30min on ice
- Heat-shock cells at 42°C during 45S (in water bath)
- Put back on ice after heat-shock
- Add 900 µL pre-warmed SOC (37℃)(rich medium)
- Incubate cells at 37°C with agitation during at least 30min
- Centrifuge cells at 4000rpm during 2min, remove 800µL of supernatant
- Plate the rest from PUC 19 positive control on LB Carb plate

- Plate 100µL from negative control respectively on LB Chloramphenicol, LB Kanamycin, LB Carbenicillin and LB Spectinomycin plates
- Incubate at 37°C overnight

Efficiency should be around 10^7colonies/µg pUC19

Transformation with chemical competent cells E.coli

Materials

- Chemical competent cells
- Ice
- Clean DNA to transform
- SOC
- Selective agar plates

Protocol

- Thaw gently cells on ice
- Add DNA (keep cells on ice)
- Incubate 30min on ice
- Heat-shock cells at 42°C during 45S (in water bath)
- Put back on ice after heat-shock
- Add 900 µL pre-warmed SOC (37°C)(rich medium)
- Incubate cells at 37°C with agitation during at least 30min
- Plate 100 µL of transformation in selective agar plate or centrifuge cells at 4000rpm during 1min, remove 800µL of supernatant and plate the rest.
- Incubate at 37°C overnight

Note

For multiple transformation (more than 10), competent cells aliquoted in PCR strip can be used. A similar protocol is used. To adapt to large volumes, cells are incubated in SOC in 96 well plates and centrifugation is therefore performed in a centrifuge adapted for plates. Cells are plated in 6 well plates filled with 3mL of LB agar supplemented with appropriate antibiotics.

Name: Pauline Mayonove
Date: 09/08/2017
Last validation/update: 09/08/2017 PV
Team Synthetic Biology protocols

TSS Buffer Preparation

Materials

- PEG 8000 (Common powders in the JB lab)
- $MgCl_2*6H_20$ (common CBS's powders)
- DMSO (under chemical hood)
- LB medium
- Syringes
- 0.2µm filters
- 15mL sterile tube

Protocol

To make 250 mL TSS Buffer:

. Mix: 25g PEG 8000
+ 1.5g $MgCl_2*6H_20$
in 12.5mL DMSO

. Add LB to 250 mL

. Filter sterilize (0.22 µm filter) under the hood PSM

. Aliquot in 15mL sterile tubes

Name: Pauline Mayonove
Date: 09/08/2017
Last validation/update: 09/08/2017 PV

SOC Medium Preparation

Materials

- SOB medium powder (with common powders in the JB's lab)
- ddH_2O
- 50mL graduated cylinder
- Ten 100mL glass bottles
- Glucose powder (with CBS common powders)
- Syringes
- 0.2µm filters
- 50mL sterile tube
- 5mL sterile pipettes
- 1.5mL sterile centrifuge tubes

Protocol

Prepare the SOB medium:
. Dissolve 14g SOB medium into 500mL (final volume) ddH2O (28g/L final concentration).
. Aliquot the solution with the graduated cylinder: 50mL in each 100mL glass bottle.
. Autoclave them the same day and cool down.
    - If laundry room staff is unable to autoclave the same day, the small autoclave fits six bottles at a time.

Prepare 2M glucose solution:
. Dissolve 18 g glucose into 50 ml (final volume) ddH_2O and filter-sterilize into a sterile 50 mL tube under the hood PSM.
. Aliquot 0.5mL per microcentrifuge tube under the hood PSM
. Store aliquots at -20°C

Prepare SOC medium:
. Add 0.5mL glucose solution into 50mL SOB medium before use.
. Divide into 10mL aliquots with SOC medium to avoid contamination.

Annex H

BioArt

During my thesis, I implemented logic circuits in living organsisms, but I also used the biological circuits that I engineered to make art. Among the constructs that I engineered for the *B. subtilis* part toolbox, I engineered two cassettes expressing high quantities of GFP and mKate fluorescent proteins. These constructs mediated a strong expression in *B. subtilis* but also in *E. coli*, indeed, *E. coli* seems less stringent than *B. subtilis* for gene expression.

I used these two constructs to paint on Petri dishes. With the team, we also organized workshop of bacteria painting with high school students.

The following are some of my bacterial painting creations.

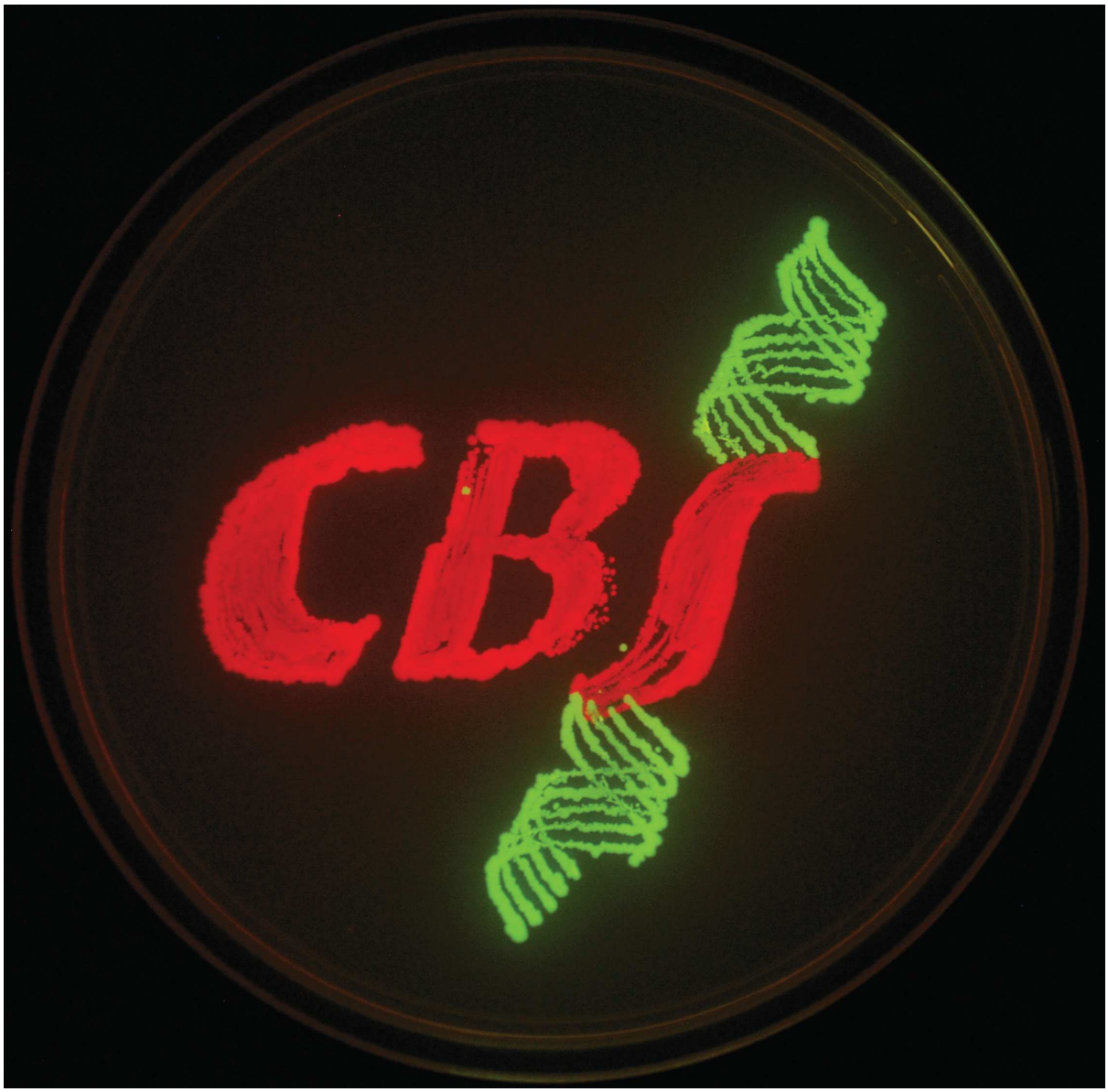


Figure H.1: **The CBS logo**

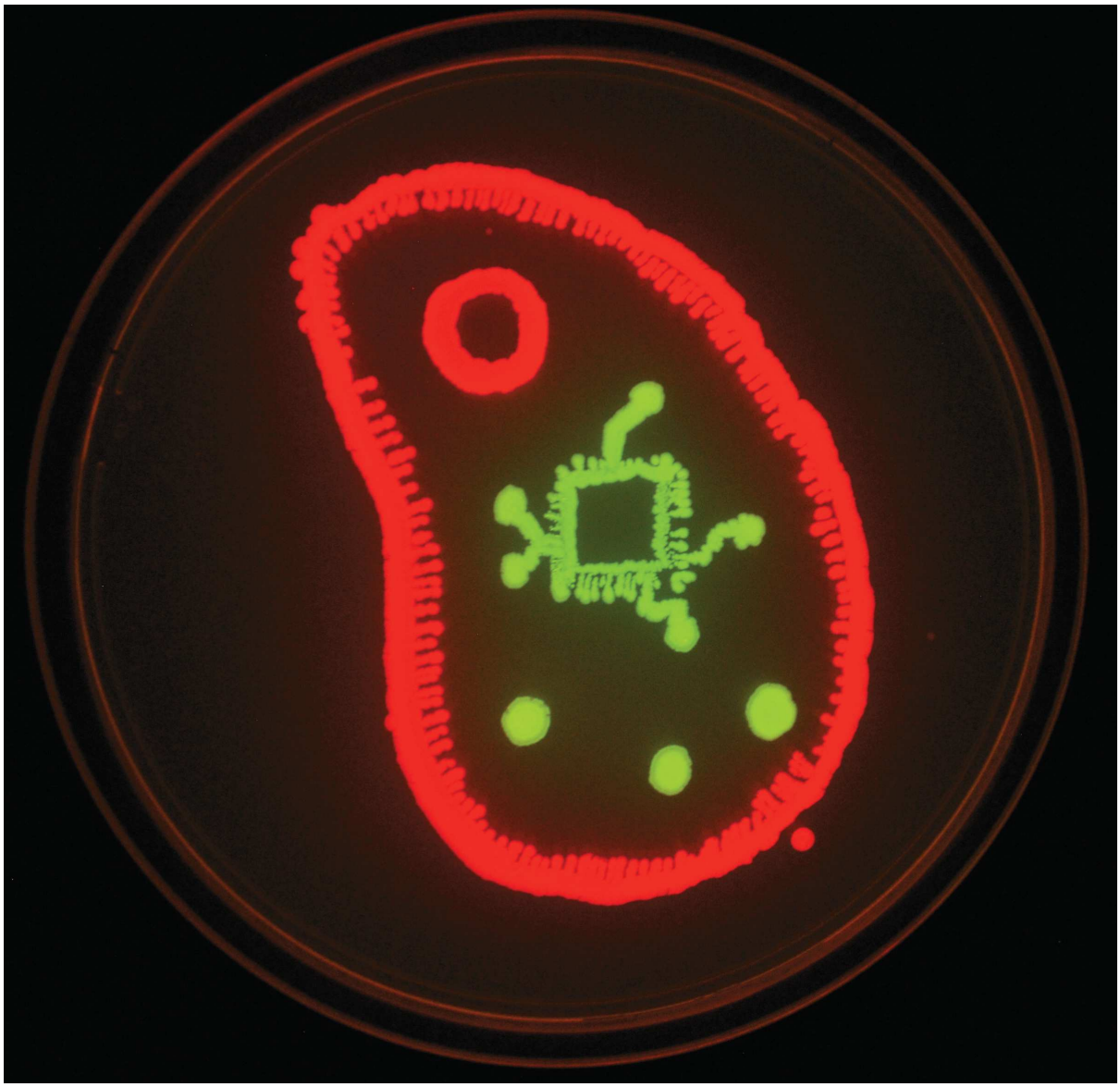


Figure H.2: **The logo of the synthetic biology team**

Figure H.3: **Dark Vador**

Abstract:

A major goal of synthetic biology is to reprogram living organisms to solve pressing challenges in manufacturing, environmental remediation, or healthcare. While many types of genetic logic gates have been engineered, their scalability remains limited. Indeed, gate design remains largely a tedious process and relies either on human intuition or on brute-force computational methods. Additionally, designed circuits are usually large and therefore not straightforward to implement in living organisms.

Here, I aimed at increasing the computation power of integrase-based logic circuits while permitting researchers to simply implement these circuits to a large range of organisms and of inputs.

First, I developed a scalable composition framework for the systematic design of multicellular systems performing integrase-based Boolean and history-dependent logic and integrating an arbitrary number of inputs. I designed multicell Boolean logic circuits in *Escherichia coli* to up to 4 inputs and History-dependent circuits to 3 inputs. Due to its scalability and composability, this design framework permits a simple and straightforward implementation of logic circuits in multicellular systems.

I also pushed forward the compaction of biological logic circuits. I generated a complete database of single-cell integrase-based logic circuits to obtain all possible designs for the implementation of up to 4-input Boolean functions. Characterization of a reduced set of circuits will have to be performed to prove the feasibility of the implementation of these circuits.

All these design strategies can be implemented via easily accessible web interfaces, and open collections of biological components that are made available to the scientific community. These tools will enable researchers and engineers to reprogram cellular behavior for various applications in a streamlined manner.

Résumé :

L'un des objectifs principal de la biologie synthétique est de reprogrammer les organismes vivants pour résoudre des challenges mondiaux actuelles dans le domaine industriel, environnemental et de la santé. Tandis que de nombreux types de portes logiques génétiques ont été conçus, leur extensibilité reste limitée. Effectivement, la conception de portes logiques reste en grande partie un processus fastidieux et repose soit sur l'intuition humaine, soit sur des méthodes computationnelles de force brute. De plus, les circuits conçus sont généralement de grande taille et ne sont donc pas faciles à implémenter dans les organismes vivants.

Durant ma thèse, mon objectif a été d'augmenter la puissance de calcul des circuits logiques utilisant des intégrases tout en permettant aux chercheurs d'implémenter simplement ces circuits à un large éventail d'organismes et d'entrées.

Tout d'abord, j'ai développé un cadre extensible et composable pour le design systématique de systèmes multicellulaires implémentant de la logique Booléenne et histoire dépendent. Ce design est basé sur l'utilisation de sérine intégrases et peut intégrer un nombre arbitraire d'entrée. J'ai implémenté dans *Escherichia coli* des circuits logiques Booléens multicellulaires jusqu'à quatre entrées et des circuits histoire-dépendent jusqu′à 3 entrées. En raison de son extensibilité et de sa composabilité, ce design permet une implémentation simple et directe de circuits logiques dans des systèmes multicellulaires.

J'ai également poussé le compactage des circuits logiques biologiques. Pour cela, j'ai généré une base de données complète de tous les circuits logiques unicellulaires possibles pour l'implémentation de fonctions booléennes à deux, trois et quatres entrées. La caractérisation d'un ensemble réduit des circuits de cette base de données devra être effectuée pour prouver la faisabilité de leur implémentation.

Je pense que ces différentes stratégies de conception et les différents outils distribués (pièces biologiques et interface web) aideront les chercheurs et les ingénieurs à reprogrammer le comportement cellulaire de manière simple pour diverses applications.