



**HAL**  
open science

# Energy-Efficient Memristor-Based Artificial Intelligence Accelerators using In/Near Memory Computing

Kamel-Eddine Harabi

► **To cite this version:**

Kamel-Eddine Harabi. Energy-Efficient Memristor-Based Artificial Intelligence Accelerators using In/Near Memory Computing. Micro and nanotechnologies/Microelectronics. Université Paris-Saclay, 2023. English. NNT : 2023UPAST086 . tel-04229739

**HAL Id: tel-04229739**

**<https://theses.hal.science/tel-04229739>**

Submitted on 5 Oct 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Energy-Efficient Memristor-Based  
Artificial Intelligence Accelerators using  
In/Near Memory Computing  
*Accélérateurs d'Intelligence Artificielle à Base de  
Memristors à Faible Consommation d'Énergie utilisant le  
Calcul Dans/Proche de la Mémoire*

**Thèse de doctorat de l'université Paris-Saclay**

École doctorale n°575: Electrical, Optical, Bio-Physics and Engineering  
Spécialité de doctorat: Electronique, Photonique et  
Micro-Nanotechnologies  
Graduate School : Sciences de l'ingénierie et des systèmes.  
Réfèrent : Faculté des sciences d'Orsay

Thèse préparée dans le **Centre de Nanosciences et de  
Nanotechnologies** (Université Paris-Saclay, CNRS),  
sous la direction de **Jacques-Olivier Klein**, Professeur,  
la co-direction de **Damien Querlioz**, Chargé de recherche,  
le co-encadrement de **Jean-Michel Portal**, Professeur.

**Thèse soutenue à Paris-Saclay, le 03 juillet 2023, par**

**Kamel-Eddine HARABI**

**Composition du jury**

<b>Dr. Julie Grollier</b> Directrice de recherche, Unité Mixte de Physique CNRS/Thales	Présidente
<b>Dr. Quentin Raffay</b> Maître de conférences, IMEP-LAHC, Université Grenoble-Alpes	Rapporteur & Examineur
<b>Pr. Melika Payvand</b> Professeur Assistante, Institute of Neuroinformatics, University of Zurich / ETH Zurich	Rapporteuse & Examinatrice
<b>Dr. Gilles Sassatelli</b> Directeur de recherche, LIRMM, CNRS/Université Montpellier	Examineur

**Titre:** Accélérateurs d'Intelligence Artificielle à Base de Memristors à Faible Consommation d'Énergie utilisant le Calcul Dans/Proche de la Mémoire.

**Mots clés:** Intelligence Artificielle, Faible consommation d'énergie, technologies de mémoire, Calcul Dans/Proche de la Mémoire, Memristors, Inférence Bayésienne

**Résumé:** L'Intelligence Artificielle (IA) émerge comme une force omniprésente dans notre vie quotidienne, possédant le potentiel de provoquer une révolution transformatrice dans une multitude de secteurs de la société. Cependant, sous cette promesse de transformation, l'IA est confrontée à deux défis majeurs qui nécessitent une attention urgente : l'efficacité énergétique et la fiabilité. Les besoins énergétiques croissants de l'industrie de l'IA contribuent aux émissions mondiales de carbone en raison des hautes exigences computationnelles des modèles d'IA, menaçant la durabilité environnementale. Parallèlement, la nature 'boîte noire' de nombreux systèmes d'IA, produisant des décisions difficiles à interpréter, soulève des questions de confiance. Ces incertitudes introduisent des risques dans des secteurs critiques, formant des barrières à l'acceptation plus large de l'IA. En réponse à ces défis, cette thèse propose une approche multidisciplinaire qui unifie l'intelligence artificielle, l'architecture informatique et les technologies émergentes. Notre stratégie implique le développement de circuits intégrés spécialisés utilisant la technologie de pointe des memristors, une technologie nanoelectronique conçue pour supporter des paradigmes de calcul à faible énergie pour les modèles d'IA, spécifiquement dans des contextes à ressources limitées. Le concept central de cette approche est d'exploiter la non-volatilité et les capacités de calcul Dans/Proche de la mémoire des memristors, tout en tenant compte de leurs caractéristiques non-idéales, pour atteindre une haute efficacité énergétique, particulièrement dans le domaine du edge computing. De plus, nous incorporons l'inférence Bayésienne, une technique d'IA totalement explicative, dans le circuit pour répondre aux problèmes de confiance associés à l'IA, favorisant ainsi le développement d'applications d'IA transparentes et fiables. Le

premier chapitre de cette thèse introduit une architecture de calcul Proche-mémoire conçue pour les applications d'IA de périphérie (AI at the Edge), inspirée par l'efficacité énergétique exceptionnelle du cerveau humain. Nous proposons une architecture de machine Bayésienne basée sur des memristors, qui ouvre la voie vers des modèles d'IA à haute efficacité énergétique. Dans le deuxième chapitre, nous explorons une machine Bayésienne qui emploie une approche de calcul stochastique au sein d'un système d'array de memristors distribué. Cette machine, que nous avons conçue, fabriquée et testée, présente une efficacité énergétique supérieure par rapport aux unités de microcontrôleurs traditionnelles pour des tâches telles que la reconnaissance gestuelle. Elle démontre une résilience aux erreurs logicielles et aux radiations, la rendant bien adaptée pour le déploiement dans des environnements rudes. Le troisième chapitre aborde les limitations du calcul stochastique dans notre machine Bayésienne et présente une solution alternative : une machine Bayésienne basée sur le calcul logarithmiques. Ce nouveau circuit, conçu, fabriqué et testé, améliore la précision et accélère les opérations d'inférence, tout en maintenant l'architecture et le design de la machine originale. Le chapitre fournit également une analyse comparative de nos machines Bayésiennes stochastiques et logarithmiques, élucidant leurs forces et faiblesses respectives. Dans le dernier chapitre, nous abordons les défis associés à l'utilisation des memristors. Nous introduisons une plateforme de prototypage basée sur des memristors multimodes qui facilite la mise en œuvre de projets analogiques et numériques. Actuellement, cette plateforme est utilisée dans deux laboratoires de recherche pour valider une gamme de concepts neuromorphiques analogiques et de logique numérique en mémoire.

**Title:** Energy-Efficient Memristor-Based Artificial Intelligence Accelerators using In/Near Memory Computing

**Keywords:** Artificial Intelligence, Low Energy consumption, Novel Memory Technologies, In/Near Memory Computing, Memristors, Bayesian inference

**Abstract:**

Artificial Intelligence (AI) is emerging as an omnipresent force in our everyday lives, possessing the potential to bring about a transformative revolution across a multitude of societal sectors. Yet, beneath this promise of transformation, AI is grappling with two significant challenges that need urgent attention: energy efficiency and trustworthiness. The AI industry's escalating energy demands are contributing to global carbon emissions due to the high computational needs of AI models, threatening environmental sustainability and restricting the deployment of AI in resource-constrained settings such as edge devices. Simultaneously, the 'black box' nature of many AI systems, producing difficult-to-interpret decisions, raises concerns about trust. These uncertainties introduce risks in critical sectors forming barriers to the wider acceptance of AI. In response to these challenges, this thesis proposes a multidisciplinary approach that unifies artificial intelligence, computer architecture, and emerging technologies. Our strategy involves the development of specialized integrated circuits utilizing cutting-edge memristor technology, a nanoelectronic technology designed to support low-energy computational paradigms for AI models, specifically in resource-constrained contexts. The central concept of this approach is to harness the non-volatility and in/near-memory capabilities of memristors, while accounting for their non-ideal characteristics, to achieve high energy efficiency, particularly in the realm of edge computing. Additionally, we incorporate Bayesian inference, a fully explainable AI technique, into the circuitry to address the trust issues associated with AI, fostering the development of transparent and

dependable AI applications. The first chapter of this thesis introduces a near-memory computing architecture designed for edge AI applications, inspired by the human brain's exceptional energy efficiency. We propose a memristor-based Bayesian machine architecture employing memristors, that paves the path towards energy-efficient AI models. In the second chapter, we delve into a Bayesian machine that employs a stochastic computing approach within a distributed memristor array system. This machine, which we have designed, fabricated, and tested, exhibits superior energy efficiency compared to traditional microcontroller units for tasks such as gesture recognition. It demonstrates resilience to soft errors and radiation, making it well-suited for deployment in harsh environments. Chapter three addresses the limitations of stochastic computing in our memristor-based Bayesian machine and presents an alternative solution: a logarithmic memristor-based Bayesian machine. This newly designed, fabricated, and tested circuit enhances precision and accelerates inference operations, while maintaining the original machine's architecture and design. The chapter also provides a comparative analysis of the stochastic and logarithmic memristor-based Bayesian machines, elucidating their relative strengths and weaknesses. In the final chapter, we tackle challenges associated with memristor utilization. We introduce a multimode memristor-based prototyping platform that facilitates both analog and digital project implementation. Currently, this platform is being used in two research labs to validate a range of digital logic-in-memory and analog neuromorphic concepts.



Thèse effectuée au sein du **Centre de Nanosciences et de Nanotechnologies**  
de l'Université Paris-Saclay, CNRS  
10 Boulevard Thomas Gobert  
91120 Palaiseau  
FRANCE

*To those who believe in the power of knowledge and  
science to shape the destiny of society for the better.*

# Acknowledgements

The completion of this Ph.D. thesis not only marks the culmination of my academic journey but also embodies a voyage enriched by the unwavering support, guidance, and inspiration of countless remarkable individuals.

Foremost, I extend my deepest gratitude to my supervisor, Dr. Damien Querlioz. I still recall our first meeting at the digital electronics course, where your enthusiasm and profound knowledge about the future of computing captivated me. It was then that I was introduced to the fascinating field of neuromorphic computing, which became the cornerstone of my research pursuits. Joining your research team, I immediately felt a sense of belonging amidst the vibrant and dynamic collaborations you had cultivated, making every day an extraordinary learning experience. Your extensive knowledge and ability to delve swiftly into new research areas brought a unique energy to our projects. I am immensely thankful for your unconditional support, close yet respectful supervision, which fostered an ideal environment for skill development and growth.

Additionally, I extend my sincere thanks to my co-supervisor, Pr. Jean-Michel Portal. your generous sharing of technical expertise in designing integrated circuits during my visits to your team in Marseilles was instrumental in sharpening my skills. Every interaction was a step forward, and for that, I am grateful. I must also express my appreciation for Pr. Jacques-Olivier Klein, another pivotal supervisor. Despite our limited meetings due to your administrative duties, I have greatly benefited from your advice, suggestions, and positive influence.

My sincere gratitude goes out to the members of my Ph.D. defense committee: Pr. Melika Payvand, Dr. Quentin Raffay, Dr. Julie Grollier, and Dr. Gilles Sassatelli. I appreciate your willingness to be part of my committee, the time devoted to studying my manuscript, and your insightful comments and rigorous feedback that have significantly elevated the quality of my work. A special mention to Dr. Belgacem Habba for accepting my invitation and contributing to the discussion with your industry-oriented questions despite the early hour resulting from the time difference. I am also thankful to all the reviewers whose insights have further refined the research articles constituting this thesis manuscript.

My journey would have been far less enriched without the steadfast support of my research collaborators, who have been integral pillars throughout this venture. Dr. Tifenn Hirtzlin and Clement Turck deserve heartfelt appreciation for being unwavering co-first authors and partners in several projects. Collaborating with Tifenn on our first project, "The Memristor-based Bayesian Machine," was an enlightening experience; your blend of motivation and diverse skills greatly enriched our collaborative moments. Not to mention your sports insights and our activities of hiking in Grenoble and playing football in Saclay. I still remember that we haven't yet settled the debate on who is faster in a running race. Clement's involvement in our

group significantly elevated the value of every research project included in this thesis. Your rapid learning and adept multitasking were truly commendable; whether conducting Bayesian inference simulations, working on PCB design, or analyzing measurements, his contributions were invaluable. Collaborating on our successful projects has been an immense pleasure, and I extend my best wishes for his upcoming Ph.D. defense. I extend my appreciation to our interns—Thibaut Loiseau and Adrien Renaudineau—for your brilliant contributions to various projects. Your grasp of concepts and tasks, often with minimal discussion, was admirable.

I extend my acknowledgment to collaborators beyond our lab's confines. My gratitude flows towards Dr. Thomas Dalgaty from CEA-Leti for involving me in the project of the in-situ Memristor-based MCMC learning; such genius ideas come from a brilliant mind like yours. Dr. Mathieu Faye from IM2NP, your support in designing the MCMC sampling machine has notably enhanced my skills in crafting complex circuits. Drs. Fadi Jebali and Eloi Muhr from IM2NP, engaging with you on various projects and sharing discussions during my visits to Marseilles were opportunities I greatly appreciated. Special thanks to Dr. Marc Bocquet from Aix Marseille University; our exchanges have always been intellectually stimulating, and I am continually impressed by your precise and critical insights during our project discussions. Gratitude is also due to Dr. Elisa Vianello and Dr. Etienne Nowak from CEA Leti for their support in design and fabrication process, ensuring the success of our tape-out projects. Additionally, I value the successful collaborations with the team from HawAI.Tech startup, including Drs. Pierre Bessière, Jacques Drouez, and Raphael Laurent, particularly on the Bayesian machines projects. Acknowledgments are also extended to Dr. Louis Hutin from Spintech lab for the MTJ-based RNG project collaboration. Every individual, with their unique expertise and spirit of collaboration, has played a significant role in expanding the breadth and depth of my research, thereby making my Ph.D. journey all the more enriching and fulfilling.

The IntegNano team, filled with brilliant minds and warm hearts, significantly enriched my professional and personal experience at our lab. My journey began alongside Drs. Atreya Majumdar, Rohit Pachat, and Xing Chen, who quickly transitioned from being colleagues to my best friends. Our myriad of activities—traveling, playing board games, practicing football, and hiking—forged countless memories, far too many to enumerate here. Your unwavering support, particularly during the challenges of the COVID-19 lockdown, was invaluable. Although I miss them already, I hold the hope that our friendship will endure. Marie Drouhin, the discussions and your talks on equilibrium propagation projects were always enlightening. Your swift understanding and effective work with the AwesomeArray chip, leading to interesting results, were truly commendable. Best wishes for your upcoming Ph.D. defense!

Joining the team, I found myself amidst talented and productive young researchers who smoothed my path and simplified myriad challenges. I extend my gratitude to the preceding generation of researchers, beginning with office-mates. Dr. Maxence Ernout, I fondly recall your invaluable assistance with implementing AI algorithms in Python, I also fondly remember the foosball games we played alongside Axel and Tifenn. Your sense of humor and camaraderie



have been truly memorable. Dr. Axel Laborieux, the wise and humble presence in our group, offered precise explanations and positive feedback that greatly benefited us all. Dr. Mamour Sarr, your serene presence and friendly football rivalry were highlights of my days. A nod of thanks also goes to Dr. Bogdan Penkovsky; our shared wu-shu classes, despite it resulted to occasional late arrival at meetings, added a unique flavor to our interactions. Acknowledgment is also due to the newer doctoral students, including Mohammed Akib, with whom I had the chance to collaborate on the "p-bit for Ising machine" project. Our work was stimulating, and I foresee you designing remarkable p-bit circuits and obtaining fascinating results from our developed chips. Working with Théo Ballet on the placement and routing of a RISC-V processor was a valuable opportunity to delve into the physical design of a complex digital system. Théo, I am confident that your contributions will elevate our research projects to new system-level heights.

A heartfelt thanks to all colleagues and friends at C2N with whom I've had the pleasure to interact. Dr. Liza Herrera Diez, your generosity and vibrancy were refreshing, and I thoroughly enjoyed the delightful treats "Chocolates and bread" to our team and beautiful paintings you shared with us. Dr. Tanvi Bhatnagarschoeffmann, your positive vibes brought light to our team, despite the relentless food talks with Atreya! Dr. Gyan Vdjagt, organizing group activities made our time more enjoyable, Dr. Maryam Massouras, our engaging discussions and your valuable advice for my Ph.D. defense were much appreciated. Dr. Guanda Wang, our hiking adventures and your generous donations during the pandemic's initial wave showcased your kind heart. Asma Mouhoub and Amina Djemmah, your support during pivotal moments, including my Ph.D. defense, was invaluable. A nod to Djohan Bonnet, Bastien Imbert, Naimul Hassan, Thomas Bersani-Veroni, and Adrien Pontlevy; although your time in the lab was brief, our intellectual exchanges enriched my experience.

I extend my gratitude to all those working at C2N who contribute to the smooth functioning of the lab. Kudos to Christophe Chassat and Alain Péan from IT, Lydia Andalon, Lea Lemaitre, Carole Bonot, Laurence Sidibé, and Bernadette Laborde from the administration, and Sophie Bouchoule and Emilia Davodeau from the doctoral school. A special mention to ChatGPT for refining the English texts for my thesis, significantly enhancing the clarity and coherence of my writing.

Journeying through the intricate paths of academia, I've come to realize it's far from a solo venture. My Best friends Dr. Bassem Boukhebouze, Khaled Saadi, and my brother Abdelkarim Harabi have been unwavering pillars of support, anchoring me with their belief and strength from the very start. A heartfelt tribute to all family and friends, especially my parents; my father, a symbol of hard work and passion, and my mother, who cultivated my fascination for science and shared her aspirations with me. Their sacrifices, love, and steadfast faith have been the foundation of my journey. My thanks go to Dr. Pietro Ferreira from University of Paris-Saclay, for teaching me essential skills in circuit design and layout, and to all the devoted teachers who have been beacons of light at every stage of my academic path.

# Contents

<b>Introduction</b>	<b>1</b>
<b>1 The Case for Building Bayesian Machines with Memristors</b>	<b>11</b>
1.1 In/Near Memory Computing with Memristors . . . . .	13
1.1.1 The Evolution of making Efficient Chips . . . . .	13
1.1.2 Toward Non-Von Neumann Machines . . . . .	24
1.1.3 Memristors for Energy-Efficient Computing . . . . .	28
1.2 The Concept of Near-Memory Compute Architecture for a Bayesian Machine . .	41
1.3 Steps of the Bayesian Machine projects . . . . .	45
1.3.1 Design Flow for Memristor-based Chips . . . . .	46
1.3.2 Measurement Setup for Memristor-based Chips . . . . .	49
1.3.3 Task Implementation and Energy analysis . . . . .	51
1.4 Comparison of our Bayesian Machines with Other Nanotechnology-Based Machine Learning Accelerators . . . . .	53
1.4.1 Our Bayesian Machines vs. Nanodevice-Based Neural Networks . . . . .	53
1.4.2 Other Bayesian Concepts Involving Nanodevices . . . . .	55
1.5 Conclusion . . . . .	57
<b>2 A Stochastic Bayesian Machine</b>	<b>59</b>
2.1 Bayesian Inference with Stochastic Computing . . . . .	61
2.2 Design of the Stochastic Bayesian machine . . . . .	64
2.2.1 The Big Picture . . . . .	64
2.2.2 Programming methodology of the Bayesian machine . . . . .	66
2.2.3 Reading strategy and Read disturb on the Bayesian machine . . . . .	71
2.2.4 Inference Using a Bayesian machine . . . . .	73
2.3 Characterization of the Stochastic Bayesian Chip . . . . .	75
2.3.1 Forming, Programming, and Read-Disturb Experiments . . . . .	75
2.3.2 Bayesian Inference Experiments . . . . .	77
2.3.3 The search for Optimal LFSR seeds . . . . .	78
2.4 Energy efficiency of the Stochastic Bayesian machine . . . . .	80
2.5 Nanodevice-Based True Random Number Generation . . . . .	84

2.5.1	Random Number Generation with MTJs . . . . .	84
2.5.2	Randomness Sensing With Precharge Sense Amplifier . . . . .	86
2.5.3	Design of SMTJ-Based RNG . . . . .	88
2.6	Conclusion . . . . .	92
2.6.1	Energy Estimation of a scaled stochastic Bayesian machine . . . . .	96
<b>3</b>	<b>A Logarithmic Bayesian Machine</b>	<b>99</b>
3.1	Bayesian Inference with Logarithmic computing . . . . .	101
3.2	Design of a Logarithmic Bayesian Machine . . . . .	105
3.3	Measurements on the Logarithmic Bayesian machine . . . . .	109
3.4	Energy Efficiency of the Bayesian Machines . . . . .	113
3.5	Large Scale Multi-computing Mode Bayesian machine . . . . .	115
3.6	Conclusion . . . . .	117
<b>4</b>	<b>Multimode Memristor-based Prototyping Platform</b>	<b>119</b>
4.1	Imperfect Memristors for Building New Computing Paradigms . . . . .	121
4.1.1	Non-Ideal Behavior of OxRAM-Based Memristor . . . . .	121
4.1.2	Mitigating Imperfections for Non-Conventional Computing . . . . .	123
4.1.3	Embracing Imperfection for Non-Conventional Computing . . . . .	125
4.2	Description of the Hybrid CMOS/Memristor Die . . . . .	128
4.2.1	Digital Mode Circuitry . . . . .	130
4.2.2	Analog Mode Circuitry . . . . .	131
4.2.3	Design Signoff and Measurement Setup . . . . .	132
4.3	Uses of the Platform . . . . .	134
4.3.1	Digital Prototyping Projects . . . . .	134
4.3.2	Analog Prototyping Projects . . . . .	135
4.4	Conclusion . . . . .	137
	<b>Conclusions and future work</b>	<b>139</b>
	<b>List of publications</b>	<b>147</b>
	<b>Bibliography</b>	<b>169</b>

# List of Figures

- 1 From Biological Neuron to Artificial Neural Network.** **a** Biological neuron (adapted from wikimedia). **b** An electrical circuit model for a neuron proposed by Hodgkin and Huxley, using basic electric circuit elements to implement bio-neuron behavior. **c** Artificial Neuron model (or perceptron), proposed by McCulloch-Pitts, the sum of the multiplication of the elements of an input vector  $X$  and a synaptic weight vector  $w$  is output by the neuron (followed by a non-linear function). **d** Artificial neural network with hidden layer, two inputs and two outputs. . . . . 3
- 2 Growth in AI compute power demands over the past six decades.** Plot of the computational power required by benchmark AI models, measured in PetaFlop-days (One petaFLOPS-day is the number of computations that could be performed in one day by a computer capable of calculating a  $10^{15}$  floating point operations per second). Models for several applications: vision, language, speech, and game models. Two different eras of progress can be distinguished based on the usage of growth slopes. In the first era, compute doubled every two years; in the second era, every 3.4 months [1, 2] (adapted from [3]). . . . . 4
- 3 Growth in AI models parameters size, and the AI dedicated hardware memory size, from 2016 to 2021.** Growth of total number of parameters that a model needs over time. The plot shows the count for state-of-the-art models in computer vision (blue points), natural language processing (red points), recommender systems (black points), as well as the maximum memory capacity of AI hardware (green points). (Adapted from [4]) . . . . . 5
- 4 Equivalent carbon-dioxide footprint for training AI on image recognition task.** The computing resources and energy required to train the best objects recognizing deep-learning systems designed for error levels at human performance (less than 5 percent in this graph) would be enormous, leading to the emission of as much carbon dioxide as New York City generates in one month [5] (adapted from [6]). . . . . 6

5	<b>Ph.D. Thesis Infographic.</b> During my thesis, I have been incorporated mainly or partially in nine research projects, resulting in six publications (see list of publications) and the design of seven emerging nanoelectronic-based integrated circuits ( 1, 2, 3, 4, and 5 are RRAM-based, 6 is MRAM-based, and 7 is FRAM based). The numbers stand for the projects, the color code is yellow for taped-out design (sent for fabrication), orange for fabricated circuits and started testing, and blue for paper publication. Most of the designs are fabricated in a hybrid 130nm CMOS-Nanodevice process; only design 7 is based on a hybrid 22nm FDSOI-Nanodevice process. . . . .	10
<b>1</b>	<b>The Case for Building Bayesian Machines with Memristors</b>	<b>11</b>
1.1	<b>Brain-like Computing.</b> The confluence of advancements in AI algorithms, hardware technologies, and nanodevices contributes to the emergence of neuromorphic computing, offering potential solutions to prevailing challenges in AI, such as energy efficiency. . . . .	12
1.2	<b>Transistors population: from one device to trillion device on a chip. a</b> The first transistor, developed by Walter Brattain and John Bardeen in 1947. (source: Nokia USA Inc. and AT&T Archives) <b>b</b> Apple M2 Chip (released on 2022), an ARM-based system on a chip (SoC) designed by Apple Inc. The M2 is made with TSMC’s FinFet Enhanced 5-nm technology, and it contains 20 billion transistors. The M2 Max version contains 67 billion transistors (Source: Apple website). <b>c</b> Wafer Scale Engine Two (WSE-2) chip, designed by Cerebras Systems, The Wafer Scale Engine (WSE) is a single, wafer-scale integrated circuit processor, it is designed for AI training and inference workloads in data-centers. The WSE-2 has 850,000 cores with a total of 2.6 trillion transistors, made with TSMC’s FinFet 7-nm technology (Source: Cerebras website). . . . .	14
1.3	<b>The Evolution of the modern world’s most important invention: the transistor. a</b> 75th Transistor anniversary (cover image of the IEEE Spectrum magazine, by Lisa Sheehan). <b>b</b> The evolution of MOSFET Based transistors (Source: Samsung Tech Blog). <b>c</b> The Dennard scaling stopped around 2005, Moore’s law trend might follow the same destiny (Reproduced from [7]). . . . .	15
1.4	<b>Photolithography from Rubylith to EUV. a</b> Hand drawing patterning on Rubylith photomasks (Source: Intel and Computer history museum). <b>b</b> The basic steps of the lithography process include substrate preparation, photoresist application, mask alignment and exposure, development, etching or deposition, and photoresist removal. <b>c</b> ASML EUV machine (Source: ASML website) . . . . .	16
1.5	<b>Evolution of chip design complexity.</b> From only functional design and verification of transistor level circuits, to multi-process multidisciplinary design and verification of Heterogeneous chips (Reproduced from [8]). . . . .	18

<p>1.6 <b>Current used Computer architecture.</b> <b>a</b> A Simplified CPU architectures, and <b>b</b> a GPU architectures (adapted from the NVIDIA documentation). <b>c</b> Spatial Architecture for Highly-Parallel Computing, suited for NPUs, it has a tiling architecture, consisting Parallel processing elements, interconnected by network on chip (Reproduced from [9]). <b>d</b> Data reuse schemes, used in most of Spatial Architecture based NPUs, for decreasing data movement by Minimizing weights movement with weight stationary scheme, Minimizing outputs movement with output stationary scheme, or Minimizing activation’s movement with activation stationary scheme (Reproduced from [9]). . . . .</p>	<p>20</p>
<p>1.7 <b>Evolution of Multi-Chip/Chiplet Packaging (Reproduced from [8]).</b> . . . . .</p>	<p>22</p>
<p>1.8 <b>Data movement and the von Neumann bottleneck.</b> <b>a</b> (Bottom) In the conventional von Neumann architecture, the memory unit and the processing unit are physically separated; the data needs to be constantly shuttled through them via a bus. This imposes a limitation in terms of speed and energy of computation: it is called the von Neumann bottleneck. (Top) A worker, company and house analogy. <b>b</b> The energy costs of single arithmetic operations for different precisions, and energy of memory access to SRAM and DRAM in a modern computer (reproduced from [10]). The energy for accessing DRAM is four orders of magnitude time higher than performing 8-bit addition operation. . . . .</p>	<p>24</p>
<p>1.9 <b>Specialized memory hierarchy for a spatial architecture.</b> <b>a</b> An example of Memory Hierarchy of a spatial architecture (Reproduced from [9]). <b>b</b> The Energy cost of data movement in a memory hierarchy (Reproduced from [9]). . . . .</p>	<p>25</p>
<p>1.10 <b>Tward Non-Von-Neumann Architecture, the brain inspired architectures.</b> <b>a</b> Near-Memory computing architecture, with worker, home and company analogy. <b>b</b> In-Memory computing architecture, with worker, home and company analogy. . . . .</p>	<p>26</p>
<p>1.11 <b>Samsung’s processing-in-memory (PIM) solution.</b> A hardware solution to accelerate the AI computation. <b>a</b> A System on Chip hardware includes GPUs and <b>b</b> high bandwidth memory (HBM-PIMs), which consist of <b>c</b> stacked DRAM based processing in memory dies (PIM-DRAM), each contain two Programmable Computing Unit (PCU), located near to 8 DRAM banks, with extra digital periphery (Source: Samsung Website). . . . .</p>	<p>27</p>
<p>1.12 <b>Early Memory Technologies.</b> <b>a</b> Punch cards, a mechanical memory that encoded data through the presence or absence of holes in specific positions. <b>b</b> Magnetic drum memory, a non-volatile memory that functioned by magnetizing small spots on a metal drum. The polarity of the magnetized spot would represent binary 0s and 1s. <b>c</b> Magnetic-core memory, used tiny magnetic toroids, the “cores”, which could be magnetized in one of two directions, representing a 0 or 1. . . . .</p>	<p>28</p>

1.13 <b>Semiconductor-Based Memory Technologies.</b> <b>a</b> A 6T SRAM cell consists of two CMOS inverters connected back to back. <b>b</b> A DRAM cell comprises a capacitor C that serves as the storage node, which is connected in series to a FET. <b>c</b> Floating gate transistor. The storage node of a flash memory cell is a floating gate of a FET, and can be used for <b>d</b> flash NOR structure or <b>e</b> flash NAND structure. (Reproduced from [11]) . . . . .	29
1.14 <b>3D integration of Memory Technologies.</b> <b>a</b> 3D stacking of DRAM based HBM memories (Source: AMD website). <b>b</b> 3D NAND Memory from Micron (Source: Micron website) <b>c</b> Empire State Building (3D urban architecture) under construction. (Source: reddit) . . . . .	30
1.15 <b>Memory Hierarchy and Addressing the Gap with Emerging Technologies.</b> . . . .	33
1.16 <b>Emerging Memory Technologies.</b> <b>a</b> Resistive RAM structure and its <b>f</b> current-voltage (I-V) characteristics for a bipolar switching device. <b>b</b> Phase-change memory structure and its <b>g</b> resistance change characteristics. <b>c</b> Magnetic RAM structure and its <b>h</b> resistance-voltage characteristics. <b>d</b> Ferroelectric RAM structure and <b>e</b> Ferroelectric FET structure and their <b>i</b> polarization–voltage hysteretic characteristic. (Reproduced from [11]) . . . . .	34
1.17 <b>In-Memory computing with Memristor crossbars for artificial neural network.</b> <b>a</b> Neural network with three inputs and two outputs mapped on a memristor crossbar of three rows and two columns. The multiply-and-accumulation operation can be performed in the analog regime, taking advantage of Ohm’s Law and Kirchhoff’s Law. <b>b</b> A memristor crossbar used as a vector-matrix multiplier, including ADCs, DACs, and digital input and output circuitry. This crossbar is a main element in the ISAAC architecture hierarchy, used to build <b>c</b> the In-Situ Multiply-and-Accumulate block that is part of <b>d</b> the ISSAC Tile block. (Reproduced from [12]) . . . . .	37
1.18 <b>In-Memory computing with resistance summation for artificial neural network.</b> <b>a</b> Resistance summation crossbar array architecture. <b>b</b> Time-domain readout method. A lumped capacitor and distributed parasitic capacitors in the array are charged, and the time taken for the voltage at the end of the column to reach a reference voltage is measured, correlating to the column resistance. The resistance value represent the dot product of the input vector and the weight vector. <b>c</b> Bit-cell structure, which combines two parallel paths, each comprising a resistive device and a MOSFET in series. <b>d</b> Implementation of an analog XNOR operation, the multiply operation for the binary neural network, by the bit-cell structure. (Reproduced from [13]). . . . .	39

1.19	<b>General architecture of the Bayesian machine.</b> The likelihoods are stored in likelihood memory arrays implemented by memristor arrays. Observations from the real world choose the appropriate probability values from likelihood memory arrays, based on which the probability values are read from likelihood arrays, which are multiplied by multipliers. At the output, the generated results encode the posterior distribution. . . . .	43
1.20	<b>Architecture of the Bayesian machine with non-conditionally independent observations.</b> This architecture performs non-naive Bayesian inference following eq. 1.5, by pooling observations $O_2$ and $O_3$ into the same column. . . . .	44
1.21	<b>Affiliation of collaborating research entities in the Bayesian machine project.</b> Including C2N, IM2NP, CEA-Leti, ISIR, and HawAI.tec. Along with image of our paper on the cover of Nature Electronics Journal [14]. . . . .	45
1.22	<b>Overview of an Integrated Circuit Design Flow.</b> <b>a</b> Computer-aided steps performed with EDA Tools. At the end, GDS mask layouts are obtained, ready for the fabrication process (Reproduced from [15]). <b>b</b> Diagram of the main steps for making a chip from system specification to ready-to-use chip (Reproduced from wikimedia). . . . .	47
1.23	<b>Diagram illustrating our custom-developed automated design flow for integrating mixed digital and memory circuits.</b> . . . . .	48
1.24	<b>Packaging or probe testing of Bayesian machine dies.</b> <b>a</b> Non-Packaged batch (Logarithmic chip) and <b>b</b> a zoom-in on one die. <b>c</b> The custom-made 25-pads probe card, used within the probe station to connect the pads of the non-packaged dies to SMA connectors. <b>d</b> The operation of connecting the 25 micro-probes to the 25 chip pads. <b>e</b> Packaged die with a J-Lead Ceramic Chip Carrier of 52 pins (JLCC52) and <b>f</b> a plastic leaded chip carriers (PLCC) sockets, a chip carrier used to form connections between packaged chips and PCB. Chips can be easily exchanged or removed. . . . .	50
1.25	<b>Measurement setups for Bayesian machine systems.</b> <b>a</b> Setup for the packaged dies. <b>b</b> Setup for the non-packaged dies. . . . .	51
1.26	<b>The different steps of a project with the Bayesian machine, from training to on-chip inference.</b> <b>a</b> Diagram summarizing the main steps of a project involving the Bayesian machine, from Bayesian model building to using the Bayesian machine to perform on-chip inference. . . . .	52



- 
- 2.1 **Architectures of adders and multipliers.** Basic architecture of **a** a floating-point adder and **b** a floating-point multiplier. Both images are reproduced from [16]. **c** A simple multiplexer can perform the sum in stochastic computing, the output is  $z = px + (1 - p)y$ . If  $p = 1/2$ ,  $z = (x + y)/2$ . **d** A logical AND gate can perform the stochastic multiplication between two bit-streams. (Reproduced from[17]) . . . . . 61
- 2.2 **General architecture of the Stochastic Bayesian machine.** Optimization of the Bayesian machine for hardware. Random numbers (RND) are generated using linear feedback shift registers (LFSRs), shared by column, and converted using digital “Gupta” circuits to a series of random bits proportional to the appropriate probability. Additionally, the likelihoods are normalized by the maximum likelihood value of the column to maximize the convergence speed of the machine. The stochastic multiplication is implemented by a single-bit AND gate. . . . . 62
- 2.3 **Fabricated memristor-based Bayesian machine.** **a** Optical microscopy photograph of the Bayesian system die. **b** Detail of the likelihood block, which consists of digital circuitry and memory block with its periphery circuit. **c** Photograph of the 2T2R memristor array. **d** Scanning electron microscopy image of a memristor in the back end of line of our hybrid memristor/CMOS process. All subfigures use consistent color codes. . . . . 64
- 2.4 **The designed memristor-based likelihood circuit.** **a** Schematic of the likelihood block presented in Fig. 2.3b. **b** Schematic of the differential precharge sense amplifier used to read the binary memristor states. **c** Principle of complementary programming of the 2T2R bit cell memristors. . . . . 65
- 2.5 **Programming circuitry for the likelihood memory arrays.** **a** Detailed schematics of the likelihood memory array, with its programming and reading periphery circuitry, displaying the voltages needed to perform a SET operation on the first row, last column left memristor R. **b** Schematics of the 2T2R bit cell connections to the reading and programming circuitry. Two level shifters (conventional level shifter LS and three-state level shifter TLS) and one sense amplifier (PCSA) are implemented in each column. One level shifter is implemented in each row. The digital signal BLEN allows choosing between the reading or programming mode. **c** Transistor-level schematics of the level shifter (LS) and **d** the three-state level shifter (TLS) circuits. . . . . 69

2.6	<b>Programming methodology for the 2T2R bit cell.</b> <b>a</b> Voltages that need to be applied on the bit line BL, bit line bar BLb, and source line SL for forming, programming a zero, and programming a one in a 2T2R Bit cell. <b>b</b> Programming voltage levels and timings used for the Forming, RESET, and SET operations (mentioned in the Methods section of the main article). <b>c</b> Table summarizing the configuration of programming signals (level shifters) for the different programming operations supported by the memory array (forming, programming a zero, and programming a one). . . . .	70
2.7	<b>Read operation for the likelihood memory array: Precharge phase.</b> <b>b</b> The read scheme involves a precharge and a discharge phase. <b>a</b> Schematic and <b>c</b> circuit simulation of the precharge phase. BL and BLb are charged to VDD. . . . .	71
2.8	<b>Read operation for the likelihood memory array: discharge phase.</b> <b>a</b> Schematic and <b>b</b> circuit simulation of the discharge phase, BL and BLb are discharged to GND with different speed. . . . .	72
2.9	<b>Read operation for the likelihood memory array: Precharge phase.</b> <b>a</b> Schematics and <b>b</b> circuit simulation of the outputs of the PCSA converging to a stable state, while BL and BLb are completely discharged to GND. <b>c</b> Experimental measurement of the read disturb on likelihood memory arrays, with a VDD value of 1.2 volts. Even after 5.7M read operations of the whole array, no error is seen. . . . .	72
2.10	<b>Detailed operation of the Bayesian machine.</b> <b>a</b> Schematic illustrating the detailed architecture of a likelihood elementary block. <b>b</b> Flowchart of the different operations to perform a Bayesian inference computation in the Bayesian machine. <b>c</b> Time diagram illustrating the operation of the Bayesian machine. . . . .	73
2.11	<b>Measurements of the fabricated memristor-based Bayesian machine.</b> <b>a</b> Measurements of the likelihood stored in the memristors, before they have been formed. As the bits are programmed in a complementary fashion involving two memristors, the result of the measurement appears random. <b>b</b> Measurements of the likelihood stored in the memristors, after they have been formed and programmed. No bit error is seen. . . . .	76
2.12	<b>Measured output of the Bayesian machine.</b> <b>a</b> measured posterior probability as a function of the expected value from Bayes' law. The different points correspond to random observation inputs. The different rows are pooled in the same graph. The points are obtained with various supply voltages VDD ranging between 0.5 and 1.2 volts. This graph is obtained with non-optimal LFSR seeds. The measured probabilities are obtained by averaging the experimental measurements over the full LFSR period (255 cycles). <b>b</b> Same as <b>a</b> , using optimal LFSR seeds. The symbols indicate which row of the Bayesian machine was used (circle, up triangle, down triangle, square: first, second, third, and fourth row). . . . .	77

2.13 **Correlations of the random numbers generated by four LFSRs.** Each graph presents the output of one of the four LFSRs of the Bayesian machine, as a function of the output of another LFSR. Each graph contains 255 points corresponding to the 255 cycles of operation of the Bayesian machine. Graphs on the diagonal (LFSR1/LFSR1, LFSR2/LFSR2, LFSR3/LFSR3, LFSR4/LFSR4) appear as  $x=y$  lines, by definition. **a** The random numbers generated with initially randomly chosen seeds used in in Fig. 2.12a. The presence of very discernible patterns in some of the graphs (LFSR1/LFSR3, LFSR1/LFSR4), indicates the existence of a strong correlation between the output of some LFSRs. On the other hand, the outputs of some LFSRs appear largely uncorrelated (LFSR1/LFSR2, LFSR2/LFSR3, LFSR2/LFSR4). The seeds for the four LFSRs are, respectively, in hexadecimal representation: 50, E9, 10, and C6. **b** The random numbers generated with the optimal seeds used in Fig 2.12b. The results show an absence of evident correlation between all outputs of the different LFSRs. The seeds for the four LFSRs are, respectively, in hexadecimal representation: EB, FB, 7E, and 5C. . . . . 79

2.14 **Application of the Bayesian machine on a practical gesture recognition task.** **a** Setup with inertial measurement unit used to record the gesture recognition dataset. **b** Masks of the placed-and-routed Bayesian machine design used to perform the design-level gesture recognition analysis. . . . . 80

2.15 **Energy analysis of the Bayesian machine on a gesture recognition task.** **a** Energy consumption of the system (Dynamic consumption and memory arrays) during the three phases of computing: loading the seeds into the LFSR, reading the memories, and the actual inference of 255 cycles. **b** Energy consumption of the system's important points during the inference phase for 255 cycles. All energy numbers are given for a supply voltage of 1.2 volts. . . . . 81

2.16 **Energy and recognition accuracy analysis of the Bayesian machine on a gesture recognition task.** **a** Mean accuracy according to the number of cycles in the inference for two types of computation: using a “power conscious” method by taking into account only the first one out for the decision (in red) and using the conventional stochastic computing by using the maximum number of one out for the decision (in blue). The shadows around the graph show one standard deviation of the mean accuracy over the ten subjects. **b** Energy consumption during the inference phase as a function of the accuracy for gesture recognition for the two methods. The stars correspond to the same point in both graphs **a** and **b**. All energy numbers are given for a supply voltage of 1.2 volts. . . . . 82

2.17 **Stochastic superparamagnetic tunnel junctions.** **a** Representation of the bistable magnetic states, and the associated low energy barrier(adapted from [18]). **b** Experimental resistance trace and thresholding operation (Reproduced from [19]). . . . . 85

2.18	<b>Comparison table between conventional Digital, Probabilistic and Quantum computing paradigms.</b> Each column shows the specifications and a simple illustration of the computational paradigms, based on basic computational units: respectively, the bit, the p-bit and the qubit. (Adapted from [20]). . . . .	86
2.19	<b>Hardware structure for TRNG.</b> <b>a</b> Simplified hardware structure for TRNG with uniform distribution. <b>b</b> Simplified hardware structure for P bit generator. (Adapted from [21]). . . . .	87
2.20	<b>Sensing circuitry for Stochastic magnetic tunnel junctions.</b> <b>a</b> PCSA based sensing circuit for reading the state of a SMTJ (Adapted from [19]). <b>b</b> a circuit for MTJ based P bit (Reproduced from [18]). . . . .	88
2.21	<b>Schematics of the designed TRNG and P-bit prototypes.</b> <b>a</b> Schematics of the first prototype TRNG circuit based on two SMTJs and a PCSA. <b>b</b> Schematics of the second prototype TRNG circuit based on two SMTJs, two biasing transistors and a PCSA, it can function as P-bit as well. . . . .	89
2.22	<b>Schematics of a designed P-bit prototype, and the XOR whitening technique.</b> <b>a</b> Schematics of the third prototype TRNG circuit based on two SMTJs, two biasing transistors and a PCSA, it can function as P-bit as well. <b>b</b> Schematics of the XOR2 whitening circuit using two basic RNG circuits and one XOR2 circuit. <b>c</b> Schematics of the XOR4 whitening circuit using four basic RNG circuits and three XOR2 circuits. . . . .	90
2.23	<b>Classification table of the designed prototypes.</b> The table lists and classify the designs . First row for the basic designs and second row for the designs with the XOR whitening technique. and Based on the type of the design. One column per prototype. . . . .	91
2.24	<b>The SMTJ Based RNG and P-bit demonstrator chip.</b> <b>a</b> Layout view of the chip with the seven designs, using two sets of 25 IO pads. <b>b</b> Optical microscopy photograph of the fabricated die. . . . .	92
2.25	<b>Suggested integration of TRNG and P-bit to the Bayesian machine.</b> <b>a</b> Schematics of likelihood with embedded TRNG. <b>b</b> Schematics of likelihood with embedded P-bit, the likelihoods are stored in analog values, the analog read of memory outputs analog biasing voltages corresponding the stored probability. . . . .	94
<b>3</b>	<b>A Logarithmic Bayesian Machine</b>	<b>99</b>
3.1	<b>Quantization functions.</b> <b>a</b> Linear quantization of probabilities represented by an unsigned four-bit fixed point format. <b>b</b> Logarithmic quantization of probabilities represented by an unsigned four-bit fixed point format. <b>c</b> Logarithmic quantization of probabilities represented by an unsigned one's complement four-bit fixed point format. . . . .	102

3.2	<b>Determining the Logarithmic Quantization Parameters.</b> This Figure explores the impact of two parameters on quantized probabilities: <b>(a)</b> the logarithmic base and <b>(b)</b> the $m$ parameter. Increasing either parameter leads to a better representation of high probability values at the cost of reduced precision and increased minimum encoded probability value. . . . .	103
3.3	<b>General architecture of the Logarithmic Bayesian machine.</b> . . . . .	105
3.4	<b>Detailed operation of the logarithmic Bayesian machine.</b> <b>a</b> Schematic illustrating the detailed architecture of a Log-likelihood elementary block. <b>b</b> Flowchart of the different operations to perform a Bayesian inference computation in the logarithmic Bayesian machine. . . . .	106
3.5	<b>Physical views of the memory block used in the Bayesian machines.</b> <b>a</b> Masks of the memory block with five routing metal layers. <b>b</b> Abstract view of the memory block, consisting of metal blockage masks to avoid routing above the memory block and pin masks to define the routing position of In and Out pins, and <b>c</b> the routing position of the supply voltages, which are easily distributed and routed. . . . .	108
3.6	<b>Fabricated memristor-based logarithmic Bayesian machine.</b> Optical microscopy photograph of the Bayesian system die. . . . .	109
3.7	<b>The experimental setup for on-chip measurements on the logarithmic Bayesian machine.</b> The setup includes a custom PCB to route an MCU with a probe station and several power supply sources. . . . .	110
3.8	<b>Inference measurements on the fabricated Bayesian machines.</b> Measured output as a function of expected result on the fabricated <b>a</b> logarithmic and <b>b</b> stochastic Bayesian machine. In the logarithmic case, all points for supply voltages ranging from 0.7 to 1.2 V are superimposed. . . . .	110
3.9	<b>Placed-and-Routed Logarithmic Bayesian Machine Masks for Gesture Recognition Analysis at the Design Level.</b> . . . . .	113
3.10	<b>Schematic illustrating the architecture of a multi-mode likelihood elementary block.</b> The likelihood elementary block used in the large-scale Bayesian version features two computing modes, the logarithmic and the stochastic. The mode needs to be decided before memristor programming. . . . .	115
3.11	<b>The fabricated large-scale Bayesian machine.</b> <b>a</b> Masks of the placed-and-routed large-scale Bayesian machine design sent for fabrication. <b>b</b> Optical microscopy photograph of the large-scale Bayesian machine. . . . .	116

- 
- 4.1 **Imperfections of OxRAM-based memristors.** **a** Current–voltage OxRAM characteristics for FORMING, SET, and RESET operations. An asymmetry in the SET and RESET programming voltages is seen (reproduced from [22]). **b** Progressive evolution of the resistance of two measured devices with consecutive weak RESET pulses. We see non-linearity and instability of the resistance change with consecutive applied voltage (reproduced from [23]). **c** Cycle-to-cycle programming variability in resistance states, Distribution of the low resistance state for different SET programming conditions (reproduced from [24]). **d** Cumulative distributions of OxRAM devices in eight different conductance levels, after standard iterative programming, a resistance drift can be seen (reproduced from [25]). . . . . 122
- 4.2 **Optimizing memristor programming algorithms.** **a** Optimized iterative programming algorithm. **b** Conductance cumulative probability distribution for eight distinct conductance levels programmed using the standard iterative programming algorithm. **c** Conductance cumulative probability distributions for eight conductance levels programmed using the optimized programming technique in **a**, stable resistance states read after 60s and 12h. (reproduced from [25]) . . . . . 124
- 4.3 **Analog in-memory computing with imperfect memristors.** **a** A single array row that can perform in analog fashion a dot product  $V \cdot g$  operation. **b** Probability density of the cycle-to-cycle variability for a single memristor. It follows a normal distribution, which makes memristors serve as a random variable. **c** (left) Gradient-based learning algorithms iteratively compute the derivative of an error metric with respect to a conductance model  $g$ , multiplied by a learning rate  $\alpha$ , to determine updates to be applied to the  $g$  parameters. The ideal memristor device should be capable of high precision and linear conductance updates. (right) The three panels show the gradient descent algorithm for an increasing number of model updates (green crosses). From an initial model, the algorithm performs gradient-based updates until it converges to a local minimum in error. **d** In our work, memristor random conductance updates are used by the sampling algorithm to perform, local random jumps on the posterior distribution, then an approximation of this distribution is stored. . . . . 126
- 4.4 **Memory cell structures.** **a** 1 Transistor 1 Resistor (1T1R) structure, for analog mode computing. **b** 2 Transistor 2 Resistor (1T1R) structure, for digital mode computing. **c** 1 Transistor 1 Resistor (1T1R) structure, for MCMC sampling. . . . . 128
- 4.5 **Fabricated Multimode Hybrid Memristor-CMOS Prototyping Platform.** **a** simplified schematic of a 1T1R cell connected to analog multiplexers, illustrating the concept of switching the access mode. **b** schematic of the hybrid Memristor-CMOS die, consisting of two-mode circuitry: analog mode (orange color) supplied by nominal voltage VDD5, and digital mode (blue color) supplied by VDD, VDDC, and VDDR. . . . . 129

4.6	<b>Digital Mode Circuitry.</b> <b>a</b> Schematic of the digital mode circuitry. It consists of address decoders, level shifters, PCSA sense and XNOR compute circuitry, input and output shift registers. The design adopts the 2T2R structure for storing one bit. <b>b</b> Schematics of the sensing circuitry with XNOR logic-in-memory feature, the pass transistor logic XNOR, the differential precharge sense amplifier used to read the binary memristor states and the SR Latch. <b>c</b> Error rate of the 2T2R approach as a function of the error rate of the 1T1R approach, in simulations assuming a perfect PCSA (black line) or experimentally measured on the integrated circuit of [26] (light blue points). Blue line: error rate of a SECDED ECC using the same number of devices as our 2T2R . . . . .	130
4.7	<b>Analog Mode Circuitry.</b> <b>a</b> Simplified schematic of 1T1R cell in analog mode, illustrating the switching of analog InOuts. <b>b</b> Schematic of the analog mode circuitry, with shift registers selecting inputs via Multiplexers, which consist of analog MUXs connected to SL, BL, and WL terminals. Each MUX is controlled by a shift register, to choose one of two analog inputs. <b>c</b> An example of measurement of memristor conductance from the memristor array. . . . .	131
4.8	<b>Fabricated Multimode Hybrid Memristor-CMOS Prototyping Platform.</b> <b>a</b> layout view, <b>b</b> Supply voltages connections and <b>c</b> Optical microscopy photograph. . . . .	132
4.9	<b>Measurements setup of the prototyping platform.</b> . . . . .	133
4.10	<b>Principle of logic-in-memory ternary input weight multiplication.</b> The input is voltage $I_N$ , the weight is stored in the 2T2R cells, the multiplication is done in two cycles. . . . .	134
4.11	<b>Measurement of memristor resistance as a function of number of RESET programming pulses.</b> A characterization experiment for implementing a synaptic learning rule. . . . .	135
4.12	<b>A Perspectives plot</b> A forward-looking plot demonstrating the projected growth in our research efforts, with the X-axis representing the demonstrators' scale from device to full system circuit, and the Y-axis indicating devices' count in the circuits. The linear line signifies the evolution of our designs,our Moore's law, from existing to future projects, showcasing an anticipated increase in both scale and complexity. . . . .	145

# List of Tables

1.1	Comparison of the design choices of the Bayesian machine with leading emerging memory-based realizations of neural network hardware blocks. Abbreviations. RBM: restricted Boltzmann machine. MAC: multiply and accumulate. PCM: Phase Change Memory. ADC: analog-to-digital converter. CDS: correlated double sampling. SLC: single-level cell. TDC: time-to-digital converter. Predet.: Predetermined. ND: not discussed. . . . .	54
1.2	Comparison of our work with approaches of the literature associating nanoelectronics with Bayesian concepts. Abbreviations. SW: software. HW: hardware. L-E: Low-Energy. NN: Neural Network. MTJ: magnetic tunnel junction. RNG: random number generation. . . . .	55
3.1	Comparison of the two Bayesian machines on the gesture recognition task. Conv: conventional stochastic computing. PC: power-conscious stochastic computing. Inf: inference . . . . .	114
3.2	Comparison between Stochastic and Logarithmic Machines . . . . .	118





# Introduction

“We can only see a short distance ahead,  
but we can see plenty there that needs to be  
done.”

---

Alan TURING

## A brief history of artificial intelligence

The transition from combustion cars to electric vehicles has become imperative due to the growing concerns over their adverse environmental impacts [27, 28]. This transition is an example of the technology development life cycle [29], a dynamic process that spans multiple stages. The cycle begins with invention, where researchers develop and explore new technologies and potential applications. Industries drive innovation and diffusion by developing, demonstrating, deploying, and adopting these technologies, continuously refining them as usage increases [30]. Eventually, less efficient or unsustainable technologies are replaced by advanced alternatives, driven by evolving human needs and regulations [31].

By contrast, some technologies undergo a continuous improvement cycle, such as electricity and internet technologies, ensuring their ongoing relevance and utility in modern civilization, others may become obsolete or replaced in favor of new technologies that are better suited to meet emerging challenges. This thesis delves into a technology that has often been likened to the “new electricity” by modern Artificial Intelligence (AI) pioneer, Andrew Ng [32]. AI is a multidisciplinary field that strives to create computational models mimicking various intelligent behaviors observed in animals, encompassing aspects such as reasoning and learning. AI has undergone a remarkable evolution since its inception, evolving from a topic of scientific curiosity to a pervasive technology in many facets of modern life [33, 34].

AI’s rich history spans multiple decades, tracing its roots back to the ideas of mathematicians and computer scientists like Alan Turing and John McCarthy [35, 36], leading to the development of sophisticated algorithms and model [33]. During the 1950s, 1960s, and 1970s, early AI research centered on symbolic reasoning and problem-solving. Pioneering programs such as the General Problem Solver (GPS) by Allen Newell and Herbert A. Simon [37] emerged in this era. Concurrently, researchers began constructing bottom-up models of nervous systems [38, 39], drawing inspiration from biological neurons and synapses (depicted in Fig. 1 a), as expounded by Alan Hodgkin, Andrew Huxley [40] (depicted in Fig. 1 b), and Hebbian learning [41]. In 1958, Frank Rosenblatt proposed the perceptron [42], based on McCulloch-Pitts neurons [39], the first general-purpose model inspired by biological neural networks (see Fig. 1 c), laying the foundation for artificial neural networks (see Fig. 1 d) and machine learning techniques like supervised, unsupervised, and reinforcement learning [43].

The 1980s and 1990s witnessed the advent of convolutional neural networks [44], and recurrent neural network architectures such as Hopfield networks [45, 46]. The back-propagation algorithm for training neural networks also emerged during this period [47, 48]. Though initially rooted in bio-inspired models, artificial neural networks gradually gravitated towards more statistical and mathematical models. Nonetheless, some researchers, like Carver Mead, persisted in exploring brain-like systems [49], giving rise to neuromorphic computing concepts in parallel [50]. By the end of the 20th century, AI research pivoted towards data-driven approaches, propelled by the rise of the internet and increased computational power, enabling the devel-

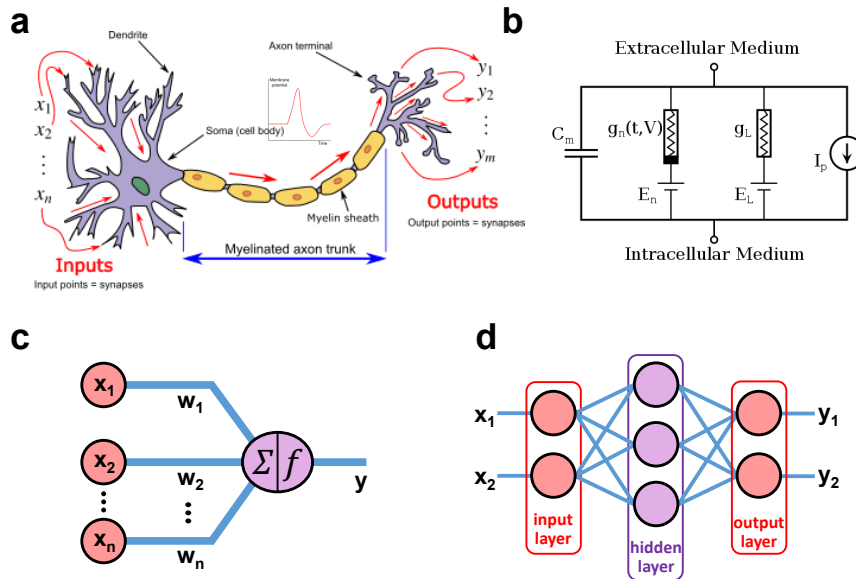


Figure 1: **From Biological Neuron to Artificial Neural Network.** **a** Biological neuron (adapted from wikimedia). **b** An electrical circuit model for a neuron proposed by Hodgkin and Huxley, using basic electric circuit elements to implement bio-neuron behavior. **c** Artificial Neuron model (or perceptron), proposed by McCulloch-Pitts, the sum of the multiplication of the elements of an input vector  $X$  and a synaptic weight vector  $w$  is output by the neuron (followed by a non-linear function). **d** Artificial neural network with hidden layer, two inputs and two outputs.

opment of sophisticated algorithms.

The 21st century has been marked by an unprecedented surge in data availability and enhanced computing capabilities of CPUs and GPUs, alongside the advent of large-capacity memories. The onset of deep learning spurred a renaissance in AI research, yielding breakthroughs in computer vision and natural language processing through deep learning architectures like Deep Convolutional Neural Networks (CNNs) and Deep Recurrent Neural Networks (RNNs) [51]. Landmark achievements in the deep learning revolution include the deep convolutional network models AlexNet [52], DeepMind's deep reinforcement learning approach with AlphaGo [53], and the protein-folding algorithm AlphaFold [54]. The development of the Transformer architecture, which employed attention mechanisms in natural language processing [55], led to considerable advancements in language understanding and text generation. This breakthrough paved the way for state-of-the-art transformer-based models such as Google's BERT [56] and OpenAI's GPT series [57], including GPT-3 and GPT-4.

## Artificial Intelligence Might Be the New Electricity, but It Has An Energy Problem

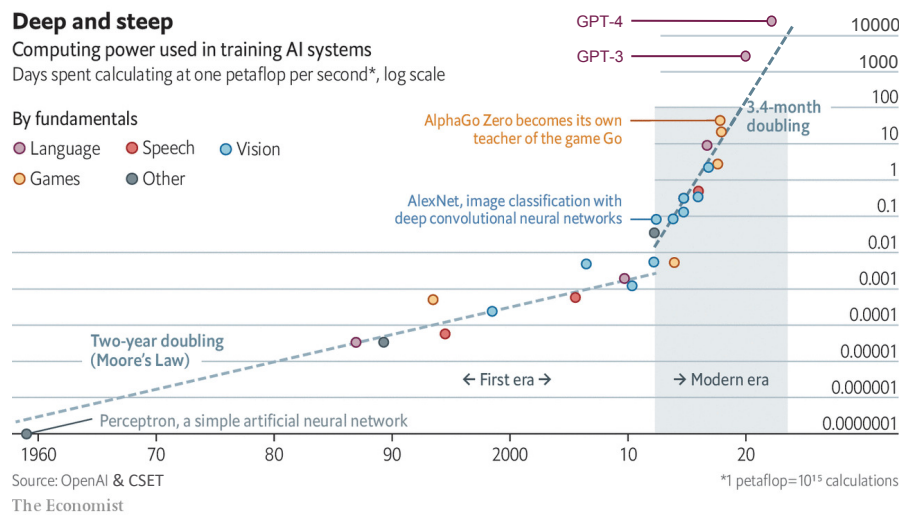


Figure 2: **Growth in AI compute power demands over the past six decades.** Plot of the computational power required by benchmark AI models, measured in PetaFlop-days (One petaFLOPS-day is the number of computations that could be performed in one day by a computer capable of calculating a  $10^{15}$  floating point operations per second). Models for several applications: vision, language, speech, and game models. Two different eras of progress can be distinguished based on the usage of growth slopes. In the first era, compute doubled every two years; in the second era, every 3.4 months [1, 2] (adapted from [3]).

In recent years, AI has transitioned into the stages of innovation and diffusion. With benchmarks like the Turing test becoming less relevant, as it arguably represents a narrow artificial intelligence test [35], large language models such as ChatGPT now exhibit early “sparks” of artificial general intelligence [58, 59]. This has led to AI’s potential being recognized by both governments and technology corporations, driving the rapid growth of its commercial applications. AI has already made significant advancements in fields such as healthcare, finance, and autonomous vehicles [60–62]. As AI increasingly becomes a vital technology in our lives, akin to electricity and the internet, it is crucial to address its fundamental drawbacks and limitations to support green and sustainable growth [63]. Although the vast availability of data and algorithmic innovations played a role in AI’s development, the rapid expansion can be primarily attributed to advancements in underlying computing hardware, particularly the utilization of GPUs [51, 52]. Since the performance of deep neural networks scales directly with their size and complexity, numerical growth is more apparent in the growing AI models in terms of parameter count and computing power [1, 2].

Examining recent advancements, shown in Fig. 2, the required computing power for deep

learning during the first era (pre-GPU era) had a growth rate in sync with Moore's law [64], doubling every two years. In the modern era, the use of GPUs has accelerated the computing process, enabling larger and more complex models. The required computing power rate now doubles every 3.4 months, posing challenges for conventional hardware (GPUs) to keep pace with this increasing demand. The memory requirement scenario for deep learning exhibits similar patterns, shown in Fig. 3. Comparing parameter counts for recent deep learning models with the volatile memory capacity (HBM and DRAMs) of modern AI-dedicated hardware reveals that, although the memory capacity of GPUs and TPUs can satisfy the requirements for computer vision models, the progress rate of GPU memory falls behind that of natural language processing models. Despite the achievements in increasing the capacity and bandwidth (HBM) of modern GPUs' volatile memories, such as the Nvidia Tesla V100 GPU, it becomes apparent that this trend will not suffice for the growing memory requirements of AI [65], indicating an approaching saturation point with these volatile memory technologies.

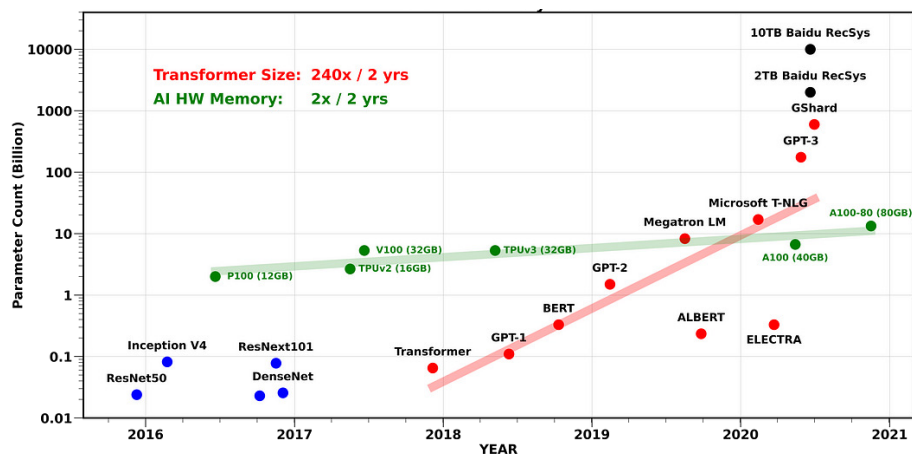


Figure 3: **Growth in AI models parameters size, and the AI dedicated hardware memory size, from 2016 to 2021.** Growth of total number of parameters that a model needs over time. The plot shows the count for state-of-the-art models in computer vision (blue points), natural language processing (red points), recommender systems (black points), as well as the maximum memory capacity of AI hardware (green points). (Adapted from [4])

The increasing demand for memory and computational power not only impacts AI development, but it also leads to higher energy consumption costs for data movement and computation [63, 66]. A significant portion of this energy is consumed during the training phase [6], which can take weeks or months using multiple energy-hungry GPUs [63, 66, 67]. For instance, To reach human-level recognition, the computing resources and energy required to train a modern deep learning model have an estimated carbon dioxide footprint equivalent to New York City's monthly emissions [5, 6, 63] (shown in Fig. 4). Despite requiring less computation than training AI models, AI inference in the cloud is still associated with significant energy costs. This is due to the sheer number of users accessing AI services, which leads to a high

volume of inference computations, and additional computation for managing these millions of access requests, which in turn requires additional energy consumption. To compound the issue, data transmission to and from the cloud also contributes to energy costs. As a result, a significant carbon dioxide footprint for inference in the cloud due to the massive scale of usage and infrastructure required to support it. This carbon footprint will continue to grow with the increasing model size and computational complexity, presenting an unsustainable trajectory from an environmental standpoint [63].

The Turing test, while providing a simple measure of AI capabilities, falls short in considering critical aspects such as energy efficiency. Presently, most modern AI models are trained in data centers [67]. As AI progresses at a rapid pace, there is a pressing need for energy-efficient hardware solutions that can adapt to the swift evolution of AI algorithms [66]. A striking example of this disparity is AlphaGo, the AI developed by DeepMind, which triumphed over the human champion in the ancient Chinese board game Go [53]. However, it faltered in terms of power efficiency, utilizing 1,920 CPUs and 280 GPUs as opposed to Lee Sedol’s 20W brain power consumption [53, 68]. This situation directs us towards a potential solution for the AI energy problem — seeking inspiration from the human brain as a model of an efficient intelligent system. Tackling the challenges of AI’s energy consumption will not only facilitate harnessing the advantages of AI at the edge but also contribute to resolving some of the trust issues surrounding AI.

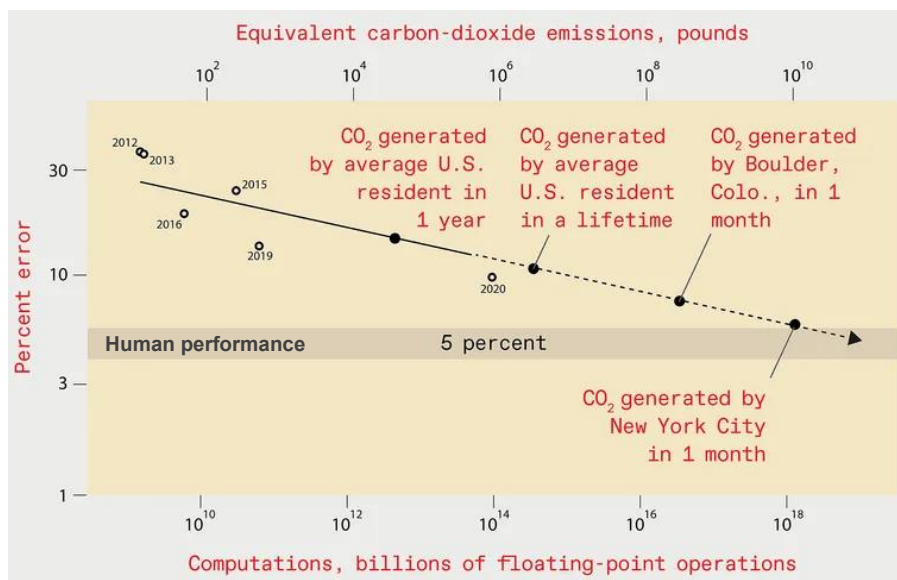


Figure 4: **Equivalent carbon-dioxide footprint for training AI on image recognition task.** The computing resources and energy required to train the best objects recognizing deep-learning systems designed for error levels at human performance (less than 5 percent in this graph) would be enormous, leading to the emission of as much carbon dioxide as New York City generates in one month [5] (adapted from [6]).

## Artificial Intelligence Also Has a Trust Problem

Intelligent systems have demonstrated immense value across various domains, yet trust issues pose significant challenges to the successful adoption and integration of artificial intelligence (AI) technologies [69, 70]. The energy costs of AI algorithms on conventional hardware have led most AI systems to upload their sensed data to the cloud for processing [71], raising privacy and security concerns [72–74]. As AI algorithms often rely on large amounts of personal data to function effectively, concerns arise regarding data collection, storage, usage, and the potential for misuse or unauthorized access.

Furthermore, deep neural networks, particularly in critical applications like intelligent medical systems, exhibit crucial limitations. First, they require massive amounts of data for training, which is often unavailable [75, 76]. Second, their results are non-explainable [77, 78], as deep learning models are often considered “black boxes” due to their complex inner workings. This lack of transparency makes it difficult for users to trust AI-generated outputs or recommendations, rendering them unacceptable for certain critical applications due to ethical and regulatory reasons. Issues such as accountability and responsibility arise when AI systems make decisions or take actions, as it can be challenging to determine who should be held accountable or responsible. Another major limitation of deep neural network-based AI systems is the absence of model uncertainty quantification [79]. This lack of uncertainty quantification can lead to challenges when deploying AI models in real-world applications, where making reliable and informed decisions is crucial. For instance, large language models like ChatGPT tend to make up or “hallucinate” responses even if they don’t have a correct answer.

Addressing these trust concerns is essential for building user confidence in AI systems and ensuring their ethical and responsible use across various domains. Researchers, policymakers, and industry leaders are working on various initiatives to tackle these concerns. Notably, AI pioneers Yoshua Bengio and Geoffrey Hinton have expressed concerns about AI regulation [80]. Bengio participated in an initiative calling for a halt in developing AI models beyond GPT-4 until proper regulations are in place, while Hinton left Google to freely discuss AI dangers and risks [81].

To tackle the trust problem in AI, several potential solutions have been proposed. Among them, Bayesian deep learning and Bayesian inference stand out as promising approaches that can enhance explainable AI, interpretability, uncertainty estimates, and robustness to data limitations [82–84]. Another compelling strategy to address trust issues in AI is the implementation of AI at the edge [85], as opposed to traditional cloud-based AI. AI at the edge offers numerous advantages, such as safeguarding sensitive information, enhancing system responsiveness, and adapting AI systems to specific users or devices. By providing these benefits, AI at the edge can help cultivate trust in AI applications.



## Summary of the thesis

The objective of this thesis is to explore solutions to the energy and trust challenges of AI by engaging in interdisciplinary research in the fields of artificial intelligence, computer architecture, and emerging technologies to develop specialized integrated circuits using novel nanoelectronic technology. This work involves designing and testing proof-of-concept AI circuits using cutting-edge memristor technology, which can support low-energy computing paradigms to implement AI models for applications in resource-constrained settings at the edge. The key focus is on exploiting the non-volatility and in-memory and near-memory capabilities of memristors while considering their other non-ideal characteristics. This approach leads to designs associating tightly logic and memory, resulting in a high energy efficiency that is well-suited for edge computing, addressing safety and privacy concerns associated with cloud-based systems. Furthermore, the incorporation of Bayesian inference – a fully explainable AI technique – into the circuitry addresses the trust issue in AI, promoting transparent and reliable AI applications.

To achieve this, during my thesis, I have been involved in nine research projects, mainly or partially, in collaboration with teams from C2N, CEA Leti, IM2NP, and Spintec. These projects explore several solutions, such as the implementation of near-memory memristor-based Bayesian inference machines [14, 86], the development of memristor-based prototyping platforms for analog and digital computing [87], in-situ learning of Bayesian models using memristors' intrinsic properties [88], energy-efficient implementation of memristor-based Binary Neural Networks and Ternary Neural Networks [89, 90], and the development of nano-device-based true random number generators. During my thesis, I have participated in the design of seven emerging nanoelectronic-based integrated circuits (including Resistive RAM, Magnetoresistive RAM, and Ferroelectric RAM), with three of them successfully fabricated and experimentally verified, three others fabricated and currently in the packaging stage, and one still in the fabrication stage. Most of these designs are based on a hybrid 130nm CMOS-Nanodevice process, and these circuits serve as proof-of-concept, ranging from prototypes of circuits with several nano-devices to a core with around a quarter-million transistors (a thesis infographic is shown in Fig. 5).

This thesis is organized into four chapters, focusing on the projects that successfully completed all phases, including design, fabrication, testing, and publication. The other projects are mentioned, in this thesis but not reported in detail.

**Chapter 1** summarizes why in/near-memory computing using memristors can be a solution to the AI energy problem and lists some state-of-the-art approaches in this context. It also describes the main principles and design choices for the memristor-Bayesian machines that we have developed during my thesis.

**Chapter 2** reports on a memristor-based stochastic Bayesian system with a hybrid CMOS/memristor chip design, fully distributed memory, and minimal data movement. It details the design, fabrication, and characterization, highlighting significant energy efficiency improvements in

a hand gesture recognition task compared to standard Bayesian inference implementations. This work is published in the Nature Electronics journal [14]. In addition, the chapter explores improving the energy efficiency of stochastic Bayesian machines using low-energy stochastic nano-devices, such as SMTJ, for random number generation. Prototype circuits based on MTJ devices were designed and fabricated for random number generation.

**Chapter 3** explores logarithmic computing in Bayesian machine architecture, focusing on energy efficiency and accuracy improvements. It presents a designed, fabricated, and tested logarithmic Bayesian machine integrated circuit and its viability despite memristor imperfections. The chapter also compares stochastic and logarithmic computing in near-memory computing circuits, highlighting the potential for enhanced statistical analysis and energy efficiency across various fields. This work is published in the proceedings of the Design, Automation and Test in Europe Conference (DATE 2023) [86]. The chapter also introduces our next Bayesian machine generation: the large-scale Bayesian machine with 143k memristors and 285k transistors was designed and fabricated for real-life applications and demonstrations.

**Chapter 4** introduces an integrated circuit for prototyping memristor-based projects, featuring both digital and analog modes. This platform enables the development and testing of innovative neuromorphic concepts, addressing memristor imperfections, challenges, and potential solutions. This work is published in the proceedings of the 28th Asia and South Pacific Design Automation Conference (ASP-DAC 2023) [87].

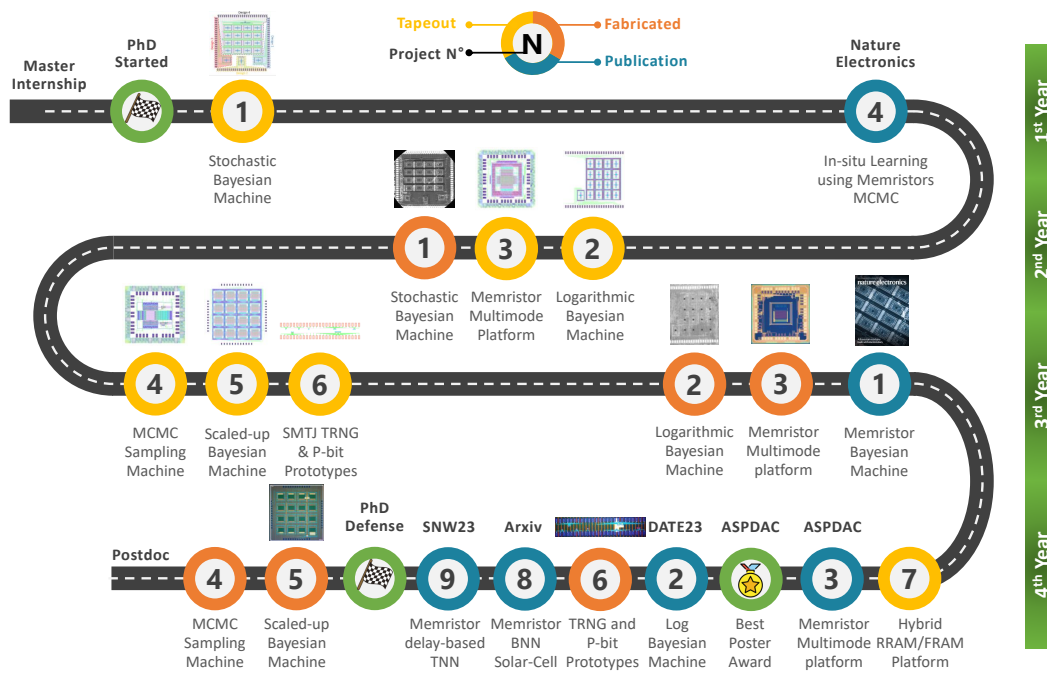


Figure 5: **Ph.D. Thesis Infographic.** During my thesis, I have been incorporated mainly or partially in nine research projects, resulting in six publications (see list of publications) and the design of seven emerging nanoelectronic-based integrated circuits ( 1, 2, 3, 4, and 5 are RRAM-based, 6 is MRAM-based, and 7 is FRAM based). The numbers stand for the projects, the color code is yellow for taped-out design (sent for fabrication), orange for fabricated circuits and started testing, and blue for paper publication. Most of the designs are fabricated in a hybrid 130nm CMOS-Nanodevice process; only design 7 is based on a hybrid 22nm FDSOI-Nanodevice process.

## Chapter 1

# The Case for Building Bayesian Machines with Memristors

Probability is orderly opinion, and that inference from data is nothing other than the revision of such opinion in the light of relevant new information.

---

Thomas BAYES

The energy and trust challenges in AI systems – including energy inefficiency in edge computing and the lack of transparency in decision-making processes – have sparked significant interest among researchers. This has prompted investigations across various domains, including AI algorithms, hardware, nanophysics, and nanodevices.

In the realm of AI algorithms, researchers are exploring lightweight neural networks, such as MobileNet [91], SqueezeNet [92], and EfficientNet [93], designed to be smaller and computationally efficient for edge devices. Quantization and pruning are other techniques employed to reduce the size and computational complexity of neural networks without significant performance loss [94], such as Binary neural networks and Ternary neural networks [95, 96]. Additionally, federated learning enables edge devices to collaboratively train machine learning models while retaining local data [97], and transfer learning leverages pre-trained models to minimize training time and computational resources needed for edge AI applications [98].

For AI hardware, AI accelerators like Google’s Edge TPU [99], NVIDIA’s Jetson series [100], and Intel’s Movidius [101], along with other AI hardware solutions based on Application-Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs), are specialized hardware components designed to enhance AI performance on energy constrained devices. In the field of nanophysics and nanodevices, memristor technologies (Resistive RAM), Magnetoresistive RAM and Phase Change Memory (PCM) and Ferroelectric RAM are being developed to offer non-volatile storage, high-density, and energy-efficient memory solutions [11, 102–104].

In this chapter, we explore the potential of Bayesian reasoning with near-memory computing architecture for AI at the edge, introducing our work on building Bayesian Machines with memristors. We begin by presenting a brief history of chip design evolution, highlighting trends that led to the choices behind our Bayesian machines. Next, we discuss the design choices involved in implementing our near-memory Bayesian machines and provide a detailed explanation of the steps required to develop a Bayesian machine with memristors. Finally, we explain how our work is positioned concerning state-of-the-art realizations of neural networks and Bayesian concepts with emerging memories.

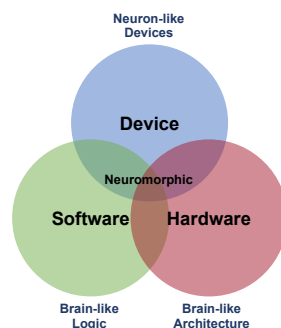


Figure 1.1: **Brain-like Computing.** The confluence of advancements in AI algorithms, hardware technologies, and nanodevices contributes to the emergence of neuromorphic computing, offering potential solutions to prevailing challenges in AI, such as energy efficiency.

## 1.1 In/Near Memory Computing with Memristors

Throughout history, humans have drawn inspiration from nature and animals to create innovative and efficient technologies using an approach called biomimicry [105]. These biomimetic inventions, ranging from Velcro (inspired by burdock burrs clinging to fur and clothing) to advancements in aviation, robotics, and sustainable architecture, have significantly impacted our civilization [106, 107]. Similarly, looking within ourselves, specifically to the human brain's natural intelligence, can guide us in ensuring the sustainable development of artificial intelligence and computing technologies [50, 108].

In this section, we delve into in/near-memory computing with memristors, a brain-inspired approach for building energy-efficient AI hardware. We start with a retrospective on chip design, emphasizing the evolution towards greater energy efficiency. Next, we explore the limitations of the traditional von Neumann architecture and the benefits of brain-inspired architectures. Lastly, we highlight the potential of memristors, an emerging technology, for energy-efficient computing, especially in the context of artificial neural network accelerators.

### 1.1.1 The Evolution of making Efficient Chips

The process of making efficient chips is akin to constructing a miniature civilization based on semiconductor devices. Over the years, this process has undergone numerous iterations and innovations, evolving from the construction of the first house, the transistor device, to the creation of villages, Small and Medium Scale Integration Circuits, and the creation of cities, Large and Very Large Scale Integration circuits, and finally, mega-cities, ultra-scale large integrated circuits, and empire on a city, System on Chip, from billions of devices up to a trillion devices (see Fig. 1.2).

The aim of chip-making is the continuous improvement of performance and efficiency, driven by advancements in chip design and fabrication. This has led to several eras, from the manual construction of chips by humans to the current era of computer-aided chip design and fabrication, and the future era of AI-assisted chip-making.

In this section, we explore the evolution of chip-making through six key pillars: transistor development, lithography and fabrication processes, design languages and tools, computer architecture, packaging technologies and sustainable creation of knowledge and talents in the field. By examining these core areas, we gain a deeper understanding of the challenges and breakthroughs that have made efficient chip-making possible.

#### 1.1.1.1 Transistor Development and Scaling

The invention of the transistor in 1947 marked a turning point in the history of electronics and computers [109, 110], as it replaced the electron vacuum tube devices (the transistor celebrated its 75th anniversary during my PhD thesis, Fig. 1.3a). Since then, continuous advancements in

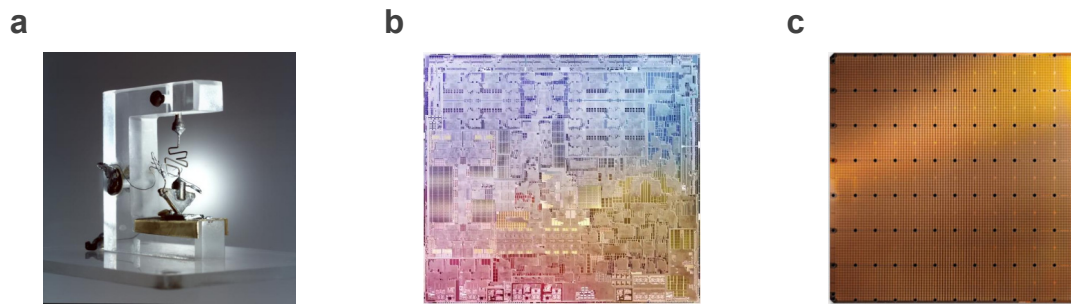


Figure 1.2: **Transistors population: from one device to trillion device on a chip.** **a** The first transistor, developed by Walter Brattain and John Bardeen in 1947. (source: Nokia USA Inc. and AT&T Archives) **b** Apple M2 Chip (released on 2022), an ARM-based system on a chip (SoC) designed by Apple Inc. The M2 is made with TSMC's FinFet Enhanced 5-nm technology, and it contains 20 billion transistors. The M2 Max version contains 67 billion transistors (Source: Apple website). **c** Wafer Scale Engine Two (WSE-2) chip, designed by Cerebras Systems, The Wafer Scale Engine (WSE) is a single, wafer-scale integrated circuit processor, it is designed for AI training and inference workloads in data-centers. The WSE-2 has 850,000 cores with a total of 2.6 trillion transistors, made with TSMC's FinFet 7-nm technology (Source: Cerebras website).

transistor design and scaling have been a major driving force behind the development of more efficient and powerful chips [111]. Early transistors made of germanium were bulky and power-hungry. The introduction of silicon as a semiconductor material in the late 1950s paved the way for the creation of smaller and more efficient transistors [112].

Transistor scaling, which involves making transistors smaller, has been a critical factor in the evolution of chip-making. It allows more transistors to be packed onto a single chip, leading to improvements in performance and efficiency. This has been particularly significant due to Moore's Law, proposed by Gordon Moore in 1965 (he passed away this year), which predicted that the number of transistors on a chip would double approximately every two years [64]. This observation has generally held true, driving the miniaturization of transistors and the corresponding increase in computational power [113].

The evolution of transistor development has seen the introduction of new transistor designs, such as the metal-oxide-semiconductor field-effect transistor (MOSFET) [114, 115]. MOSFETs have become the dominant transistor design in modern chips, allowing for faster switching speeds and lower power consumption. This transistor technology has undergone several generations of evolution. Planar manufacturing process introduced in 1959 [116], led to the invention of Planar FETs [116], showed in Fig. 1.3b, paved the way for scaling of transistor dimensions, until it reached its performance limitation at a scale of 28 nm. As Planar FETs transistor dimensions have decreased, challenges related to power consumption, heat dissipation, and leakage current have emerged, necessitating the development of innovative materials and transistor designs [114, 117].

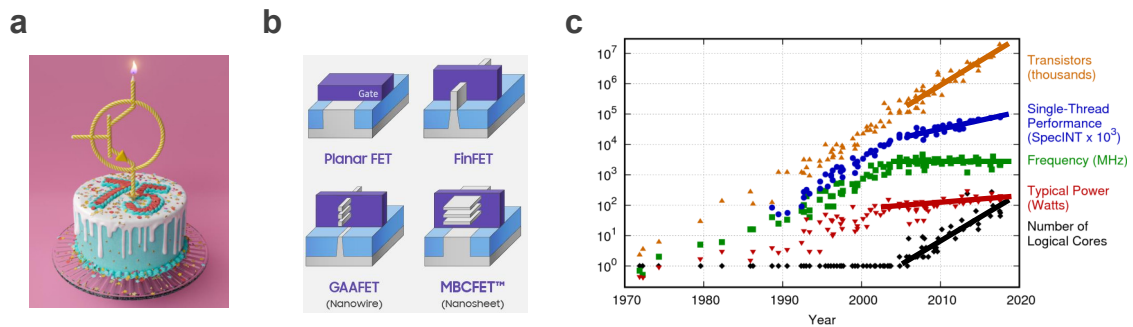


Figure 1.3: **The Evolution of the modern world's most important invention: the transistor. a** 75th Transistor anniversary (cover image of the IEEE Spectrum magazine, by Lisa Sheehan). **b** The evolution of MOSFET Based transistors (Source: Samsung Tech Blog). **c** The Dennard scaling stopped around 2005, Moore's law trend might follow the same destiny (Reproduced from [7]).

FinFETs technologies were introduced in the early 2000s, and are a type of MOSFET transistor that features a thin, vertical silicon “fin” that protrudes from the silicon substrate [118, 119], showed in Fig. 1.3b. This new transistor technology is well-suited for performance applications and has allowed scaling to continue until current days, reaching a scale of 3 nanometers, as demonstrated this year by TSMC [120]. Concurrently, FD-SOI (Fully Depleted Silicon-On-Insulator) transistors, developed by researchers at CEA-Leti, are a type of MOSFET design that uses a thin silicon layer on top of an insulating layer to reduce power consumption and improve performance [121, 122]. Making them particularly well-suited for low-power applications. To allow Moore's Law to continue, a new technologies are now under development and employment for sub-5-nm nodes. GAA-FET (Gate-All-Around Field-Effect Transistor) is a transistor design featuring a gate wrapped around a nanosheet-shaped channel (or nano-wire channels), as shown in Fig. 1.3b, allowing for better electrostatic control and improved switching behavior [123]. GAA-FETs offer several advantages, such as smaller feature size, reduced leakage current and improved speed performance. This technology has already been used this year by Samsung for their 3-nanometer node [124].

Two decades ago, there were ten champions in the transistor scaling race. However, today only two of them have reached mass production of cutting edge 3-nanometer nodes, TSMC and Samsung [125]. This achievement is not solely related to financial resources, as the USA companies have considerable resources [126, 127], or human resources, as Chinese companies have a large workforce [128]. Instead, it is primarily due to the development of a complete ecosystem surrounding this field of technology, which has been achieved by Taiwan and South Korea [129]. For the long run, there are several challenges facing the electronics industry to maintain the rapid rate of innovation and continue to follow Moore's Law [130]. The industry must overcome complex manufacturing techniques and high production costs, as well as the physical limits of transistor scaling [131, 132]. As transistors become smaller, they are more prone to various effects that can affect their behavior [132]. While there is still potential for



improvement, the margin is not as significant as it has been in the past [130]. Moore's law may not hold forever, as Fig. 1.3b illustrate, Dennard scaling already ended around 2005, and recently Moore's law is facing challenges to be kept alive.

To maintain the rapid rate of innovation in the semiconductor industry, new approaches are needed to overcome the risk of Moore's Law dying. One approach is to explore new computing paradigms such as neuromorphic computing, analog computing, in/near memory computing, etc. Another approach is to develop innovative packaging and interconnection techniques such as 3D integration and faster memories. Emerging devices and solutions such as 2D materials [133], photonic computing [134], and quantum computing may also hold promise for future developments in the semiconductor industry [135].

### 1.1.1.2 Lithography and Fabrication Processes

The construction of modern civilization required humans to transition from building simple clay houses to creating skyscrapers, roads, bridges, and supply chains by developing increasingly sophisticated techniques, materials, and tools. Similarly, to accommodate the shift from a single transistor to billions of transistors on a single chip, electronics research and industry had to innovate and refine techniques, materials, fabrication processes, and tools. One such critical technique is photolithography [136].

Lithography plays a pivotal role in integrated circuit (IC) fabrication and the transistor scaling process. This technique refers to the process of transferring a pattern from a photomask (or reticle) onto a photosensitive material (photoresist) on a substrate, typically a silicon wafer. The basic steps of the lithography process include substrate preparation, photoresist application, mask alignment and exposure, development, etching or deposition, and photoresist removal (shown in Fig. 1.4b).

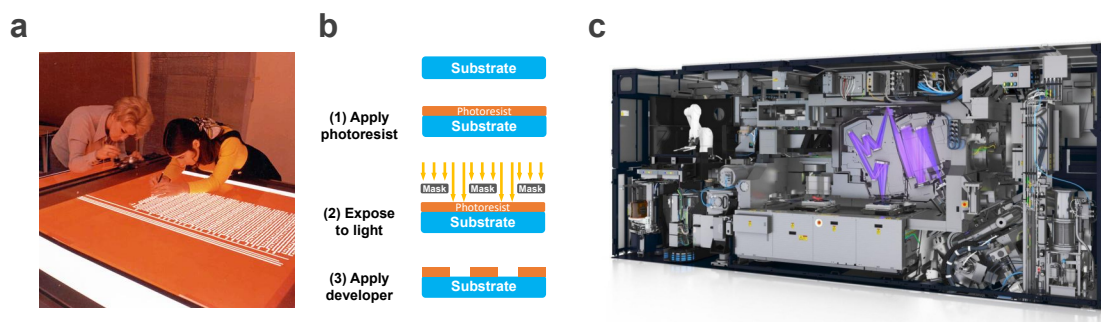


Figure 1.4: **Photolithography from Rubylith to EUV.** **a** Hand drawing patterning on Rubylith photomasks (Source: Intel and Computer history museum). **b** The basic steps of the lithography process include substrate preparation, photoresist application, mask alignment and exposure, development, etching or deposition, and photoresist removal. **c** ASML EUV machine (Source: ASML website)

The history and evolution of lithography techniques and machines can be divided into sev-

eral distinct eras, each characterized by the development and adoption of new techniques and machines that pushed the limits of transistor scaling. The earliest lithography techniques involved hand-drawn patterns on Rubylith photomasks (see Fig. 1.4a), which were then transferred to substrates using contact printing. These methods were limited by the precision of hand-drawn patterns and the resolution achievable with contact printing. Steppers, the first reduction projection systems, were introduced in the 1970s, using a reduction lens to project a smaller image of the photomask pattern onto the substrate [137]. The 1980s saw the introduction of excimer lasers, which emit deep ultraviolet (UV) light, allowing for higher resolution and smaller feature sizes [138]. In the 1990s, phase-shift masks and optical proximity correction (OPC) were developed to further improve resolution [139, 140]. The early 2000s brought immersion lithography and multiple patterning techniques, using a liquid medium between the lens and the substrate and combining multiple lithography steps to overcome the limitations of conventional lithography [141]. The most recent development is extreme ultraviolet lithography (EUV), which employs light with a wavelength of 13.5 nm [142]. It enables even higher resolution and smaller feature sizes, facilitating the continuation of Moore's Law. EUV lithography, which has been in development for several decades, is now being employed in high-volume manufacturing by only one Dutch multinational corporation, ASML, showed in Fig. 1.4c.

These advancements in lithography have gone hand in hand with improvements in the fabrication process. This process involves a series of steps, with lithography serving as a central technique. Throughout the years, new materials and innovations have been introduced to further enhance the fabrication process, such as chemical mechanical planarization (CMP) [143], atomic layer deposition (ALD) [144], and high-k materials [145]. High-k materials, with a high dielectric constant, have been introduced to replace traditional silicon dioxide gate dielectrics in transistors, reducing gate leakage current and enabling further scaling of transistor dimensions.

### 1.1.1.3 Design Languages and Design Automation Tools

Building a city is not like building a village, building a city is indeed a complex process that requires adherence to numerous rules and regulations. Similarly, the design and fabrication of complex chips with a high number of transistors also necessitate the implementation of rules, design methodologies, and advanced tools to ensure the successful creation of functional, efficient, and reliable electronic devices. Therefore, design languages (HDLs) and design automation tools (EDA) were invented and evolved in parallel with integrated circuit progress.

Hardware Description Languages (HDLs) and design automation tools, or Electronic Design Automation (EDA) tools, evolved in parallel, as the complexity of digital systems increased over time [146]. The development of both HDLs and EDA tools has been driven by the need for more efficient design, simulation, verification, and fabrication processes in the face of growing design complexity, as shown in Fig. 1.5.

In the late 1960s and early 1970s, the earliest HDLs, such as ISPS and AHDL, emerged to simplify the design and simulation of digital circuits [147]. In parallel, the first EDA tools, like SPICE for analog circuit simulation, were developed to help automate the analysis of electronic circuits [148]. As digital systems grew more complex in the mid-1970s to early 1980s, more sophisticated HDLs, like HILO and RTL/2, were developed to describe circuits at the register-transfer level [149]. Concurrently, EDA tools evolved to support schematic capture and layout design, enabling designers to create and edit circuit schematics and layouts more efficiently.

By the late 1980s, the development of VHDL, a standardized language for the design and verification of digital systems, marked a major milestone in the evolution of HDLs. Around the same time, Verilog, another widely-used HDL, was introduced. EDA tools also progressed, with simulation tools like ModelSim emerging to support HDL-based design and verification. As the adoption of HDLs like VHDL and Verilog grew, EDA tools advanced to support synthesis, place-and-route, and verification tasks. Logic synthesis tools, such as Synopsys Design Compiler, and place-and-route tools, like Cadence Innovus, started automating the process of converting HDL descriptions into gate-level netlists and optimized physical layouts.

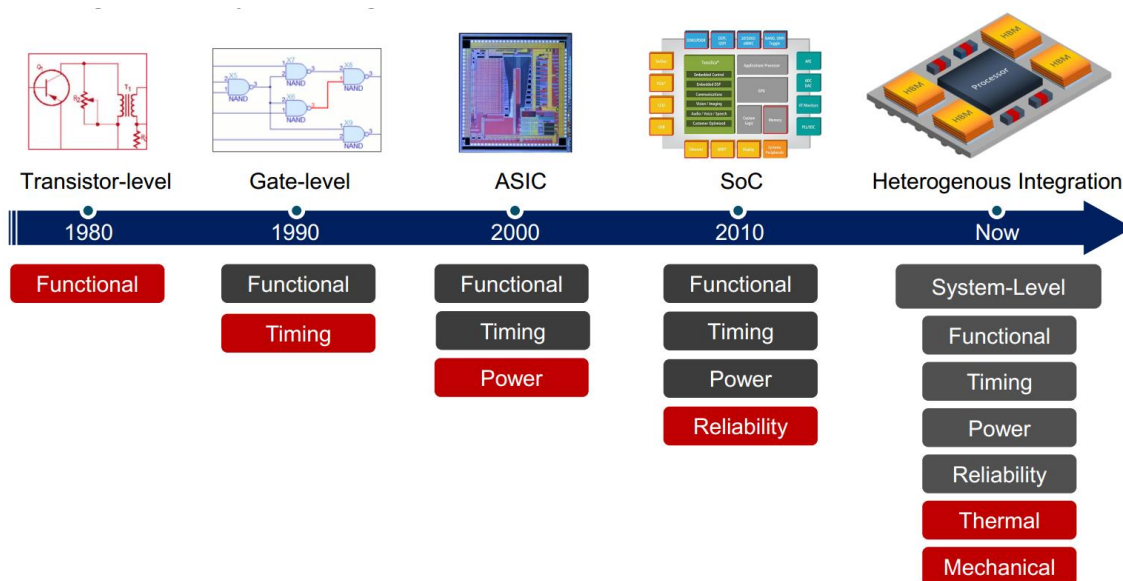


Figure 1.5: **Evolution of chip design complexity.** From only functional design and verification of transistor level circuits, to multi-process multidisciplinary design and verification of Heterogeneous chips (Reproduced from [8]).

In the early 2000s, SystemVerilog, an extension of Verilog, was introduced, offering improved support for object-oriented programming, advanced data types, and enhanced verification capabilities. EDA tools also evolved, with verification tools such as formal and functional verifiers becoming more sophisticated to ensure the correctness and reliability of designs. SystemC, a C++ extension for system-level modeling and simulation, emerged in the late 1990s and early 2000s to address the need for higher-level abstractions. High-Level Synthesis (HLS) tools

were developed to automate the process of converting high-level programming languages like C, C++, or SystemC into hardware implementations, further speeding up the design process.

From the 2000s to the present, EDA tools have continued to evolve, incorporating advanced optimization techniques, physical verification, and sign-off tools to ensure manufacturability and reliability. In parallel, HDLs and high-level design methodologies, such as HLS, have gained traction, enabling designers to handle the increasing complexity of digital systems more efficiently. Current EDA tool development addresses the increasing complexity of electronic systems and evolving industry trends. Key advancements include handling advanced process nodes, such as 3-nm and 2-nm nodes, incorporating machine learning and AI-based techniques to help optimize various design tasks [150], developing emerging computing EDA tools, enhancing hardware security mitigating potential hardware vulnerabilities, focusing on system-level design and optimization challenges, adopting cloud-based EDA, improving interoperability and IP reuse, and supporting 3D integration.

#### 1.1.1.4 Computer Architecture and Instruction Set Architecture

As cities continue to grow and expand, there is an increasing need for specialized urban planning and design, also known as city architecture. This discipline involves designing and organizing the physical layout, infrastructure, and public spaces within cities and other settlements. The ultimate goal of city architecture is to create functional, sustainable, and aesthetically pleasing urban environments that can accommodate the needs of residents, businesses, and visitors.

In the realm of computing, a similar need for organization and design arises as computing chips are complex systems, consisting of various components such as processing units, memory hierarchy, data buses, storage, and input/output (I/O) systems. This calls for a well-defined computer architecture [151], which refers to the design and organization of a computer system's hardware components that work together to process data and execute instructions. The primary objective of computer architecture is to optimize performance, power consumption, cost, and other factors for a specific set of applications or target markets. An important part of computer architecture is Instruction Set Architecture (ISA): it is an interface between the hardware and software that defines a set of instructions a computer can understand and execute. The ISA acts as a blueprint for designing the processor and the compiler that generates machine code.

The evolution of computer architecture and instruction set architecture (ISA) has been marked by continuous advancements and innovations aimed at improving computing performance, efficiency, and capabilities. In the 1940s and 1950s, early electronic computers like the ENIAC, EDVAC, and IBM 701 used vacuum tubes and employed machine-specific assembly languages [152–154]. The concept of stored-program computers emerged during this time, allowing instructions and data to be stored in the same memory. In the 1950s and 1960s, with transistors replaced vacuum tubes, IBM introduced the System/360 during this period, featur-

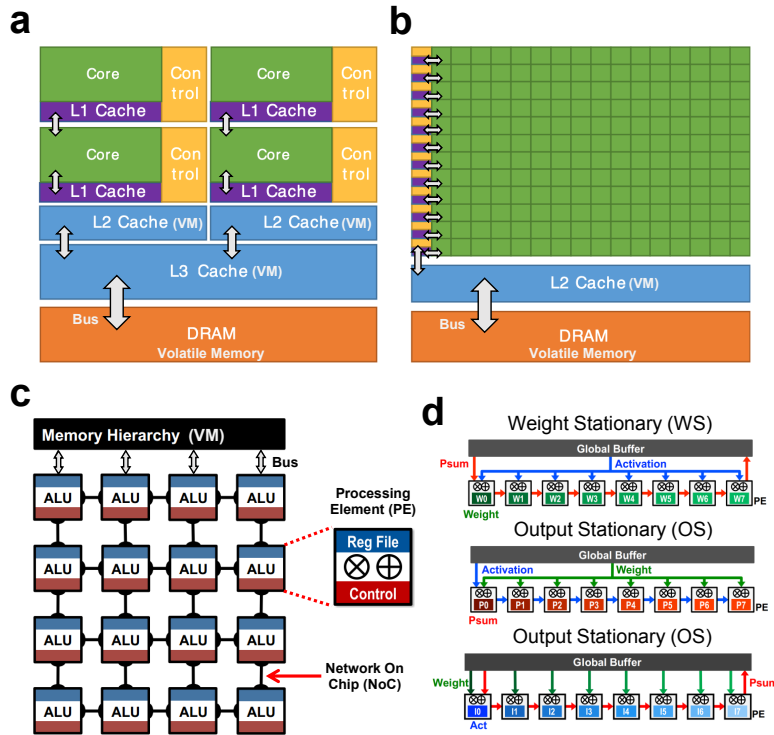


Figure 1.6: **Current used Computer architecture.** **a** A Simplified CPU architectures, and **b** a GPU architectures (adapted from the NVIDIA documentation). **c** Spatial Architecture for Highly-Parallel Computing, suited for NPUs, it has a tiling architecture, consisting Parallel processing elements, interconnected by network on chip (Reproduced from [9]). **d** Data reuse schemes, used in most of Spatial Architecture based NPUs, for decreasing data movement by Minimizing weights movement with weight stationary scheme, Minimizing outputs movement with output stationary scheme, or Minimizing activation's movement with activation stationary scheme (Reproduced from [9]).

ing a family of computers with compatible ISAs that allowed for a range of performance and cost options [155]. The 1960s and 1970s saw the rise of integrated circuits (ICs). Microprocessors, such as the Intel 4004 and 8080, emerged during this time, bringing computing power to smaller devices and setting the stage for personal computers [156].

The development of RISC architectures in the 1980s represented a significant shift in chip design, and marked the beginning of the RISC vs. CISC debate, leading to two distinct approaches to ISA design. RISC architectures, like MIPS and ARM, focused on a smaller set of simple instructions for faster execution, emphasizing simplicity and efficiency over the complexity of the previously dominant complex instruction set computer (CISC) architectures. While CISC architectures, like x86, featured more complex instructions for better memory efficiency. In the 1990s and 2000s, the era of parallelism began, with superscalar processors executing multiple instructions per clock cycle. Multi-core processors, such as the Intel Core and AMD Opteron series, became prevalent, allowing for increased performance and power efficiency. The emer-

gence of multicore processors and graphics processing units (GPUs) has further diversified the landscape of computer architectures.

In the 21st century, the focus shifted to heterogeneous computing and specialized architectures for specific workloads [157]. Graphics processing units (GPUs) gained prominence for parallel processing tasks. These developments have been accompanied by the creation later of NVIDIA's CUDA, tailored to simplify their use.

As artificial intelligence (AI) has become increasingly prominent in recent years, the need for more powerful and efficient computing platforms has grown [157]. Traditional AI implementation using Central Processing Units (CPUs) has been limited by the architecture's shortcomings, including its inability to meet AI's parallel computing and vast memory demands. Although memory caches and Dynamic Random Access Memory (DRAM), have been placed close to CPUs to increase performance in terms of speed [151], the Single Instruction, Single Data (SISD) architecture remains inadequate for AI computing (a simple illustration of CPU's architecture shown in Fig. 1.6a). The use of Graphics Processing Units (GPUs), initially designed for rendering graphics, has improved AI computing performance by accelerating processes through parallel computing [158]. While their Single Instruction, Multiple Data (SIMD) architecture has been a viable solution for data centers and AI in the cloud, it comes with significant energy costs due to communication, data exchange, and cooling requirements (a simple illustration of GPU's architecture shown in Fig. 1.6b). Additionally, GPUs occupy large areas and require extensive cooling systems, making them unsustainable for AI's growing models and incompatible with energy and area-constrained edge systems, such as wearable and battery-powered devices.

To address the energy efficiency challenges in AI hardware, specialized accelerators like Google's Tensor Processing Unit (TPU) [159], Intel Vision Processing Unit (VPU) [101], various Neural Processing Units (NPU) [66], and FPGA-based accelerators have been developed by research groups, companies, and start-ups. Most of those solutions use a spatial architecture, a tiling architecture of parallel processing elements, interconnected using a network on chip (NoC) [66] (a simple illustration of a spatial architecture shown in Fig. 1.6c). These new developments are part of a new trend toward more heterogeneous computing, involving so-called domain-specific architectures.

#### 1.1.1.5 Packaging and Advanced Packaging Technologies

The growing population and expansion of cities have made transportation increasingly challenging, as traditional road-based systems struggle with traffic congestion. In response, cities have sought innovative solutions, such as underground metros and trains, which offer a third dimension to transportation infrastructure. These systems help reduce travel time and distance within and between cities. Similarly, in the realm of modern chips, packaging plays a crucial role in enhancing performance by improving data transfer rates (interconnection bandwidth) and reducing energy consumption. By bringing components closer together, advanced

packaging techniques facilitate better performance in terms of speed, energy efficiency, and spatial footprint.

As chips have become more powerful and complex, the need for effective packaging technologies has grown. Early packaging solutions, such as dual in-line packages (DIPs) and pin grid arrays (PGAs), were adequate for simpler chips but were limited in terms of performance, thermal management, and form factor [160].

The introduction of advanced packaging technologies, such as flip-chip, ball grid array (BGA), and chip-scale packages (CSPs), addressed these challenges by enabling improved electrical performance, reduced power consumption, and smaller form factors. These packaging solutions also facilitated the integration of multiple chips or dies within a single package, leading to the development of system-in-package and package-on-package (PoP) technologies.

More recently, 2.5D and 3D packaging techniques have emerged, employing techniques like through-silicon vias (TSVs), interposers, and wafer-level packaging to achieve even greater levels of integration and performance [161]. These advanced packaging technologies have been instrumental in addressing the challenges of increased chip complexity, while also enabling the development of heterogeneous systems that integrate different types of chips, such as CPUs, GPUs, and memory, into a single package.

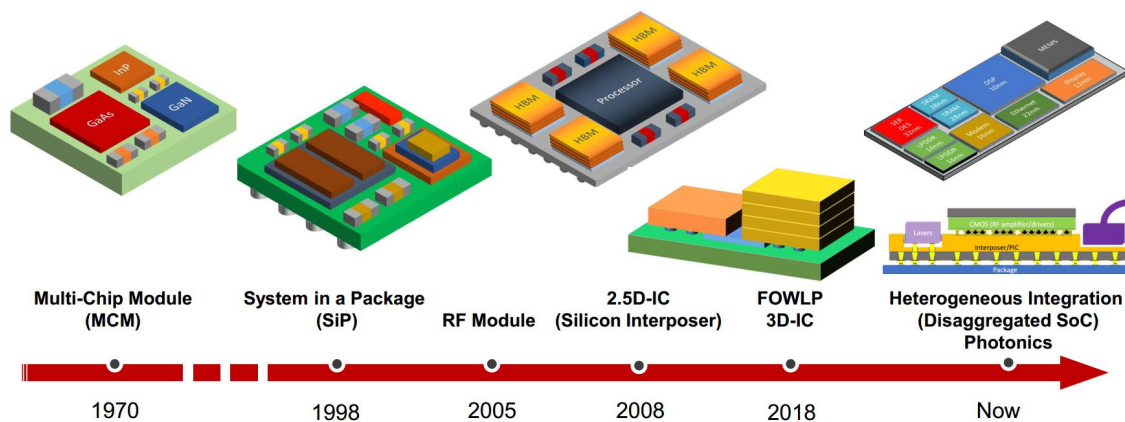


Figure 1.7: Evolution of Multi-Chip/Chiplet Packaging (Reproduced from [8]).

### 1.1.1.6 Democratized, Sustainable Chip Design

“In the advance of civilization, it is new knowledge which paves the way, and the pavement is eternal,” said Willis R. Whitney. The evolution of human civilization cannot merely be represented by modern cities; rather, it is deeply intertwined with the quality and quantity of free, shared knowledge within human societies. It also reflects our collective responsibility towards sustainable development and preserving the environment for future generations.

In the same vein, ensuring sustainable progress in the field of designing efficient chips demands increased accessibility to knowledge and resources. This encompasses a wide range of

---

opportunities, such as access to books, online courses, open-source projects, and innovative architectural solutions. Additionally, providing open access to EDA tools and Foundry PDKs through university programs, as well as offering multi-project wafer services to universities and research labs, is crucial.

The semiconductor industry currently faces a talent shortage, making it imperative to tap into the vast pool of potential talent worldwide. Talented individuals are not defined by gender, ethnicity, or nationality but can be found across diverse backgrounds and communities. By democratizing access to knowledge and resources in chip design and development, we can foster a more inclusive and dynamic ecosystem that encourages innovation and breakthroughs.

In summary for the full section, the evolution of making efficient chips has been driven by innovations and advancements across a range of domains, including transistor development, lithography and fabrication processes, design languages and tools, computer architectures and ISAs, and packaging technologies. As the industry continues to push the boundaries of chip design and fabrication, new challenges and opportunities will undoubtedly emerge, requiring further breakthroughs in these key areas to sustain the ongoing growth in chip performance and efficiency.



## 1.1.2 Toward Non-Von Neumann Machines

### 1.1.2.1 Limitations of the von-Neumann Architecture

The von Neumann architecture, named after its creator John von Neumann, has been a cornerstone in the history and evolution of computers. As the foundation for most modern computer systems, this groundbreaking architecture has profoundly influenced the computing landscape. Central to the von Neumann architecture are key components such as memory for storing data and instructions, and a processing unit comprising a control unit for managing data flow and an arithmetic and logic unit for executing computations (see Fig. 1.8a bottom).

The widespread adoption of the von Neumann architecture can be attributed to its numerous advantages and strengths, which include simplicity, general-purpose design, compatibility, and most importantly, scalability. This scalability allows for the development of more powerful computers through component upgrades without redesigning the entire system, enabling the electronics industry to concurrently develop computer components with various technologies, such as processing units with scaling MOSFET transistors and memories based on diverse technologies for specific purposes, including DRAM, SRAM, HDD, and SSD.

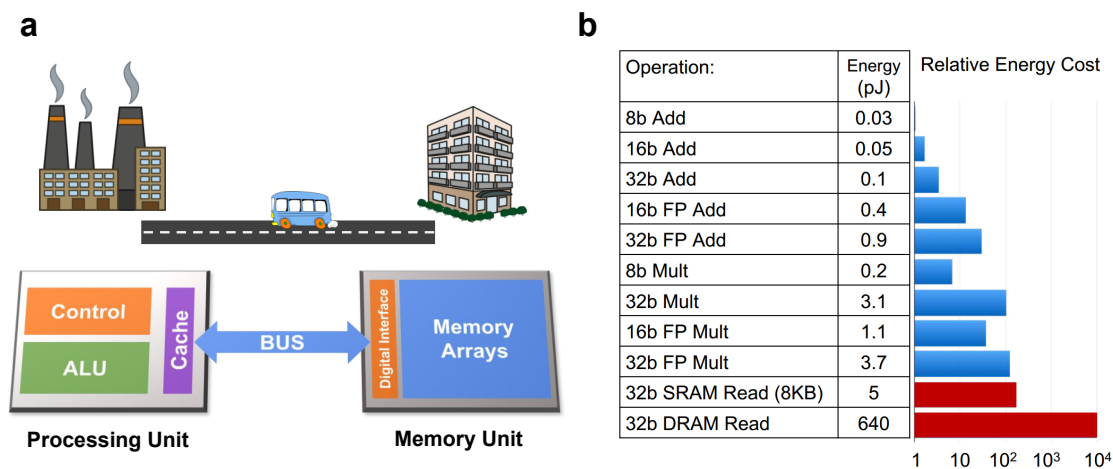


Figure 1.8: **Data movement and the von Neumann bottleneck.** **a** (Bottom) In the conventional von Neumann architecture, the memory unit and the processing unit are physically separated; the data needs to be constantly shuttled through them via a bus. This imposes a limitation in terms of speed and energy of computation: it is called the von Neumann bottleneck. (Top) A worker, company and house analogy. **b** The energy costs of single arithmetic operations for different precisions, and energy of memory access to SRAM and DRAM in a modern computer (reproduced from [10]). The energy for accessing DRAM is four orders of magnitude time higher than performing 8-bit addition operation.

Despite numerous enhancements and optimizations over the years, the von Neumann architecture has inherent limitations that hinder performance, particularly in the context of artificial intelligence (AI) and other data-intensive applications [162, 163]. The von Neumann

bottleneck, which arises from the separation of memory and processing units, is a primary source of energy inefficiency and performance bottlenecks in AI workloads (see Fig. 1.8b).

To address these limitations, specialized AI accelerators have been developed, explicitly designed to optimize AI computations. These include accelerators with spatial architectures, which employ strategies such as supporting massive parallelism to enhance computational efficiency and incorporating specialized memory hierarchies and on-chip volatile memory, including registers, SRAM, and DRAM, to reduce data movement overhead and mitigate the von Neumann bottleneck [66, 159] (see Fig. 1.9).

These accelerators yield reduced latency and improved energy efficiency for AI computations compared to traditional CPUs and GPUs. However, depending on memory hierarchies and on-chip volatile memory introduces its own set of challenges, including concerns related to area, memory density, and energy consumption. One notable limitation is the inherent volatility of memory. Accessing data from non-volatile memories (HDD or SSD) and transferring it to accelerators entails traversing the entire memory hierarchy, consuming additional energy. Moreover, using volatile cache memory with limited capacity, such as registers, results in constant data changes, as they cannot store data for extended periods while new data is processed. The general limitations of SRAM and DRAM are explored further in the following subsection.

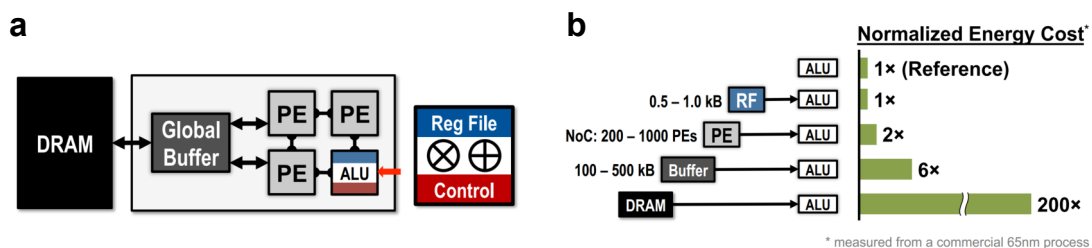


Figure 1.9: **Specialized memory hierarchy for a spatial architecture.** **a** An example of Memory Hierarchy of a spatial architecture (Reproduced from [9]). **b** The Energy cost of data movement in a memory hierarchy (Reproduced from [9]).

Current AI accelerators emphasize energy efficiency by integrating more memory into computing elements and minimizing data movements. Still, while these solutions have lessened the impact, they have not entirely solved the energy problem associated with the memory wall, as they tend to retain the “spirit” of the von Neumann architecture, with a main memory separate from the processing units.

The von Neumann architecture can be likened to having companies situated outside a city, where workers need to commute daily using buses or cars (depicted in Fig. 1.8a top). This daily transportation consumes both time and energy. While this arrangement might be suitable for spatially oriented companies, such as production, mining, and agriculture firms, it creates an inefficient system for other types of businesses. For example, service companies could benefit from being located closer to their employees and clients, thereby optimizing the overall efficiency of the system, instead of relocating their employees closer to the companies.

### 1.1.2.2 Brain Inspired Architectures, Near-Memory and In-Memory Computing

To address the limitations of the von Neumann architecture, researchers have explored alternative computing paradigms known as non-von Neumann architectures. These alternative solutions encompass a wide range of approaches, including paradigm-shifting ideas such as cellular automata, quantum computing, optical computing, DNA computing, and brain-inspired computing. By reorganizing the way data is processed, stored, and communicated within a computing system, these approaches aim to overcome the von Neumann limitations and its associated challenges.

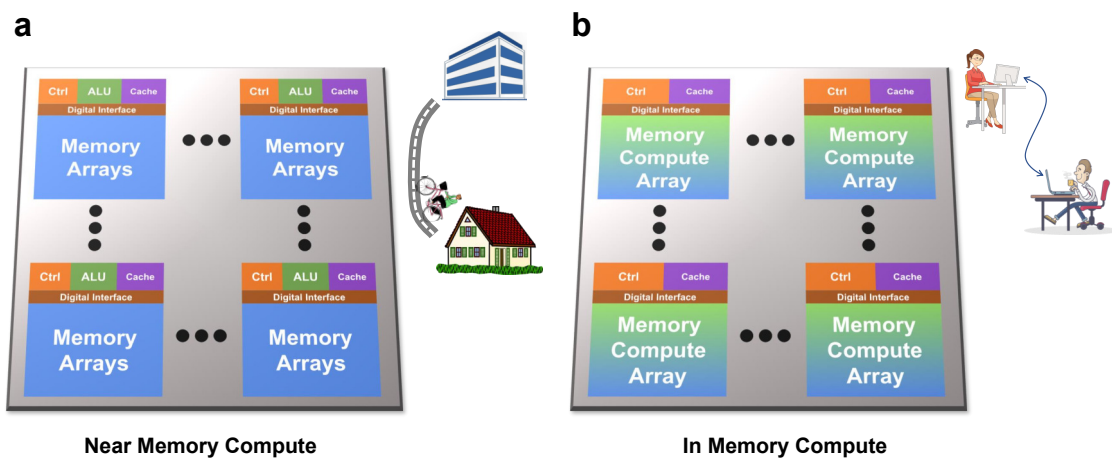


Figure 1.10: **Tward Non-Von-Neumann Architecture, the brain inspired architectures.** **a** Near-Memory computing architecture, with worker, home and company analogy. **b** In-Memory computing architecture, with worker, home and company analogy.

The pursuit of energy-efficient AI hardware has led researchers to investigate emerging computing paradigms inspired by the human brain's efficiency. Despite performing complex tasks, the brain consumes approximately 20 Watts of power, which is orders of magnitude less than current AI systems when performing advanced tasks [68]. A crucial aspect of the brain's energy efficiency lies in the collocation of computation and memory, a concept akin to in-memory computing.

Neuromorphic computing, or brain-like computing, is an emerging field that replicates various aspects of the brain's physical elements, connections, logic, architecture, and learning rules [164]. This field has given rise to multiple approaches, ranging from mimicking the brain's computing processes, such as the spike-timing-dependent plasticity (STDP) learning rule, to emulating its physical elements and connections, as demonstrated in the neuromorphic chip TrueNorth from IBM [165], and Loihi from Intel [166].

Drawing from the brain's architecture, in-memory and near-memory computing architectures have emerged to address the von Neumann bottleneck for AI and data-centric computing applications [167–169]. By closely integrating memory and processing units, these architec-

tures optimize data processing and reduce energy consumption by minimizing data movement (see Fig. 1.10).

Near-memory computing [169] places a small processing unit near the memory, reducing data movement overhead by avoiding accessing off-chip memories or using low-latency, energy-hungry buses, thus reducing energy consumption. Using an analogy of a worker and a company, the worker no longer needs to take a bus or car to go to work, as the company is located in their city (in their neighborhood or the next one), so they can walk or ride their bike to the company (depicted in Fig. 1.10a).

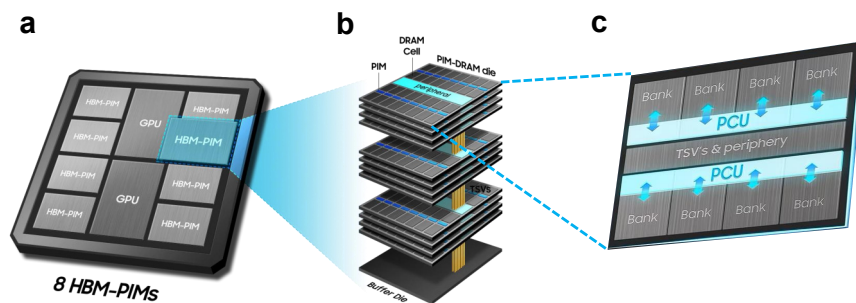


Figure 1.11: **Samsung's processing-in-memory (PIM) solution.** A hardware solution to accelerate the AI computation. **a** A System on Chip hardware includes GPUs and **b** high bandwidth memory (HBM-PIMs), which consist of **c** stacked DRAM based processing in memory dies (PIM-DRAM), each contain two Programmable Computing Unit (PCU), located near to 8 DRAM banks, with extra digital periphery (Source: Samsung Website).

Near-memory computing architecture can leverage mature memory technologies, such as SRAM and DRAM near-data processing systems, as well as emerging memory technologies like memristors. This approach has already progressed to the deployment stage, such as Samsung's processing-in-memory (PIM) products based on HBM-DRAMs, demonstrating promise for data-centric and AI applications (presented in Fig. 1.11).

In-memory computing represents a more radical paradigm shift in computing architecture, as it integrates computation within memory devices [167, 168]. This approach aims to significantly reduce data movement, with fixed memory data and only inputs and results moving. Although this architecture can leverage mature memory technologies, such as SRAMs and DRAMs, it works better with specialized memory devices, such as memristors, which involve analog storage and computation for high parallelism computing, at the cost of increased design complexity. Using the worker and company analogy again, depending on their type of work, the worker can do remote work and no longer needs to move at all. They only need to send and receive tasks and data via the internet, but they require specialized tools, such as a computer, internet connection, and VPN access (depicted in Fig. 1.10b).

### 1.1.3 Memristors for Energy-Efficient Computing

As the need for more energy-efficient and high-performance computing solutions continues to grow, the search for new memory technologies that can overcome the limitations of existing SRAM and DRAM has intensified. Memristors, a class of emerging memory technologies, have shown great potential for addressing these challenges and enabling new computing paradigms.

#### 1.1.3.1 Brief Evolution of Memory Technology

Memory technology has been a cornerstone in the evolution of computers since the dawn of the digital age. The history of memory technology is characterized by continual advancements in speed, density, and energy efficiency. The inception of the von Neumann architecture, which segregated the computer's processing and memory units, marked a significant starting point, paving the way for the independent evolution of volatile and non-volatile memory systems.

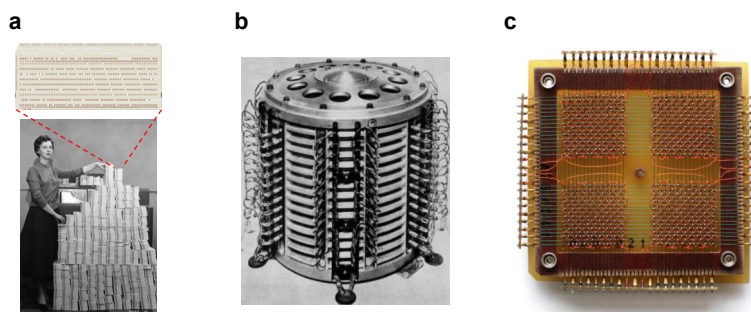


Figure 1.12: **Early Memory Technologies.** **a** Punch cards, a mechanical memory that encoded data through the presence or absence of holes in specific positions. **b** Magnetic drum memory, a non-volatile memory that functioned by magnetizing small spots on a metal drum. The polarity of the magnetized spot would represent binary 0s and 1s. **c** Magnetic-core memory, used tiny magnetic toroids, the “cores”, which could be magnetized in one of two directions, representing a 0 or 1.

The initial phase of computer memory began with mechanical systems, utilizing punch cards in devices such as Charles Babbage’s Analytical Engine in the mid-19th century (see Fig. 1.12a). The electronic computer era that emerged in the 1940s and 1950s brought significant progress in memory technology. The earliest electronic computers employed vacuum tubes to represent binary states. However, these were not a persistent form of memory and were prone to heat-induced failure. Subsequently, magnetic drum memory was introduced as a non-volatile storage medium (see Fig. 1.12b). It functioned by magnetizing specific areas on a metal drum to represent binary 0s and 1s. However, the access times were prolonged due to the drum’s rotational movement for reading or writing data. The magnetic-core memory, introduced in the 1950s, was a significant stride in memory technology (see Fig. 1.12c). It utilized small toroidal magnetic cores that could be magnetized in two directions to represent a binary 0 or 1. This technology served as the forerunner to contemporary Random Access Memory (RAM),

enabling random access to memory locations.

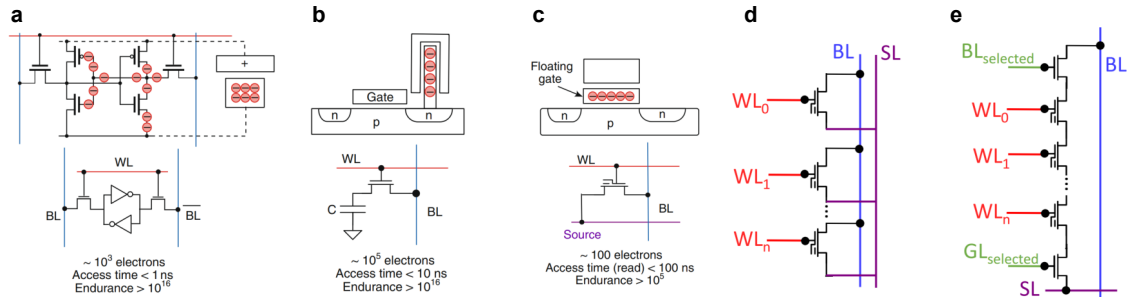


Figure 1.13: **Semiconductor-Based Memory Technologies.** **a** A 6T SRAM cell consists of two CMOS inverters connected back to back. **b** A DRAM cell comprises a capacitor C that serves as the storage node, which is connected in series to a FET. **c** Floating gate transistor. The storage node of a flash memory cell is a floating gate of a FET, and can be used for **d** flash NOR structure or **e** flash NAND structure. (Reproduced from [11])

The most impactful change occurred with the advent of semiconductor memory in the 1960s. By leveraging integrated circuits, semiconductor memory could store binary data within flip-flops or capacitors. It was during this era that the concepts of Random Access Memory (RAM) and Read-Only Memory (ROM) emerged. ROM, a non-volatile memory, stored firmware like BIOS used in the bootstrapping process, while RAM, a volatile memory, temporarily stored data for high-speed CPU access. During the 1970s, RAM underwent a significant evolution with the introduction of Dynamic RAM (DRAM) and Static RAM (SRAM). SRAM stored data using a six-transistor memory cell, resulting in a faster but more costly solution (see Fig. 1.13a). DRAM, storing data within an integrated circuit capacitor, needed periodic refreshing due to capacitor charge leakage. Despite this, DRAM became the standard for main system memory in computers, favored for its relatively low cost and smaller physical size per bit of storage (see Fig. 1.13b). The 1980s and beyond saw DRAM evolve into various forms, until the DDR SDRAM (Double Data Rate SDRAM) variant of SDRAM became the standard for modern computer systems, with successive generations (DDR2, DDR3, DDR4, DDR5) offering superior transfer rates and reduced power consumption.

The 1980s also marked the advent of Flash memory, a type of Electrically Erasable Programmable Read-Only Memory (EEPROM). This technology uses floating-gate transistors to store data, which can be electrically erased and reprogrammed, proving ideal for storage devices like USB flash drives (see Fig. 1.13c). Furthermore, hard disk drives (HDDs) were developed as a result of significant advancements in magnetic storage. These devices provided large storage capacities and were extensively used for storing operating systems, applications, and user data. Optical storage technologies, such as CDs, also made their appearance during this period, with DVDs and Blu-ray Discs following in the 1990s and 2000s, offering greater storage capacities. Flash memory took a central role in the mid to late 2000s, with Solid-State Drives

(SSDs) beginning to supplant traditional Hard Disk Drives (HDDs). SSDs use NAND-based flash memory to store data, boasting considerable advantages over HDDs, such as faster access times, lower latency, and resistance to physical shock (see Fig. 1.13e). As the cost per gigabyte for SSDs has steadily decreased, they are increasingly becoming the standard for primary storage in modern computer systems. Moreover, SSDs themselves have evolved with the introduction of NVMe (Non-Volatile Memory Express) technology, which utilizes the high-speed PCIe (Peripheral Component Interconnect Express) interface to achieve faster data transfer rates than traditional SATA (Serial ATA) interfaces.

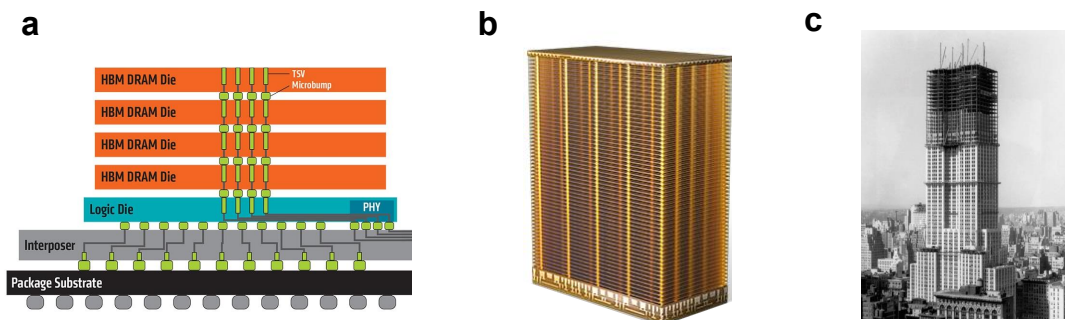


Figure 1.14: **3D integration of Memory Technologies.** **a** 3D stacking of DRAM based HBM memories (Source: AMD website). **b** 3D NAND Memory from Micron (Source: Micron website) **c** Empire State Building (3D urban architecture) under construction. (Source: reddit)

As computer capabilities have advanced, particularly in graphics processing units (GPUs), and with modern applications like artificial intelligence algorithms demanding ever-increasing data storage and memory, there has been a pressing need for memory technologies to enhance their density and bandwidth. The solution came in the form of exploiting the third dimension, much akin to the evolution in urban architecture where skyscrapers serve as a model for high-density constructions, and elevators enable rapid movement within them (example of 3D urban structure, the Empire State Building, shown in Fig. 1.14c). One of the most significant advancements in memory technology has been the development of 3D integration and 3D stacking. This methodologies entail stacking memory cells vertically, thereby enhancing density without necessitating a reduction in the size of individual cells. Within the sphere of flash memory, this innovation has fostered the creation of Vertical NAND (V-NAND) or 3D NAND technology (example of Micron 3D NAND memory shown in Fig. 1.14b). This approach stacks memory cells in multiple layers, leading to increased densities, decreased memory access power consumption, and improved reliability. In the realm of DRAM technology, the emergence of High Bandwidth Memory (HBM) marks a noteworthy progression [170]. HBM employs 3D-stacking along with Through-Silicon Vias (TSVs), vertical electrical connections that traverse an entire silicon wafer or die (See Fig. 1.14a). HBM offers significantly higher bandwidth compared to traditional DDR memory, making it ideal for use in high-performance systems such as graphics cards and high-performance computing systems. As the evolution of memory technology

continues, it remains a pivotal component in the ongoing development and enhancement of computing systems.

As we look towards the near future of memory technology, it is apparent that continued progress in 3D stacking will be a significant trend. However, as demands for greater performance and energy efficiency escalate, the limitations of current memory technologies are becoming increasingly evident. This has spurred exploration into alternative methodologies. Among these, emerging technologies such as memristors hold significant potential. These devices promise to fundamentally transform how we store and access data, thereby pushing the frontiers of computing further.

### 1.1.3.2 The Limitations of SRAM, DRAM, and Flash memories for In/Near-Memory Computing

While in/near-memory computing based on mature memory technologies has demonstrated promise in addressing the von Neumann bottleneck, there are inherent limitations associated with the use of SRAM and DRAM in these architectures. Understanding these limitations is crucial in identifying potential areas for improvement and exploring alternative approaches.

**Limitations of SRAM** Despite its advantages, such as speed, low latency, compatibility with CMOS processes, and scalability with transistor scaling, SRAM has several limitations when employed in in/near-memory computing:

*High Power Consumption:* SRAM cells consume a significant amount of static power due to leakage current, even when not actively accessed. This results in high energy consumption, which is a major concern for energy-efficient in/near-memory computing.

*High Power Consumption:* SRAM cells consume a significant amount of static power due to leakage current, even when not actively accessed. This issue is exacerbated as transistor nodes become smaller. This results in high energy consumption, which is a major concern for energy-efficient in/near-memory computing.

*Large Cell Size:* SRAM cells typically require six transistors per cell, leading to a relatively large cell size. This affects memory density, limiting the amount of on-chip memory that can be integrated within an in/near-memory computing architecture.

*Cost:* Due to the larger cell size, SRAM is more expensive to manufacture than other memory technologies like DRAM. This higher cost can be a barrier to widespread adoption in cost-sensitive applications.

**Limitations of DRAM** Although DRAM provides higher density and lower cost compared to SRAM, and faster speed than conventional NVM (such as NAND and NOR flash memories), it has its own set of limitations in the context of in/near-memory computing:

*Refresh Overhead:* DRAM cells require periodic refresh cycles to maintain stored data, which consumes energy and reduces the effective memory bandwidth available for computation.



This refresh overhead can negatively impact the performance of in/near-memory computing architectures.

*Higher Latency:* DRAM cells exhibit higher access latencies compared to SRAM cells, which can limit the performance gains achievable through in/near-memory computing.

*Complex Integration:* Integrating DRAM cells with processing units in an in/near-memory computing architecture can be challenging due to the inherent differences in their fabrication processes. This may limit the potential benefits of DRAM-based in/near-memory computing.

**Limitations of Flash memory** The idea of in-memory or near-memory computing using flash memory is intriguing because it can potentially bring computation closer to where the data is stored. The non-volatility, multi-level state (MLS) storage, high density, low cost, and low power consumption of flash memory add to its attractiveness for such applications. However, despite these appealing advantages, flash memory technology also presents significant challenges limitations:

*Very High Latency:* It is inherently slower compared to SRAM or DRAM, particularly when it comes to write and erase operations, which can significantly hamper the performance of computational tasks that require frequent data updates.

*Write Complexity:* The necessity of erasing entire blocks before rewriting complicates data management and could slow down computations.

*Higher error rates:* Flash memory is subject to higher error rates, especially when programmed in multi-level states. This impacts reliability and necessitates the implementation of complex error-correcting code mechanisms.

*Complex Integration:* Similar to DRAMs, Integrating Flash memory with processing units can be challenging due to the inherent differences in their fabrication processes.

In conclusion, while SRAM, DRAM, and Flash memory technologies each offer viable solutions for in/near-memory computing, their applicability depends largely on specific use cases. For example, in/near-memory computing based on DRAM can be used to accelerate AI computations, offering marginal improvements in energy efficiency for applications that can tolerate higher energy consumption. Conversely, in/near-memory computing that utilizes SRAM could potentially replace register files in the spatial architecture of Neural Processing Units (NPU), thereby reducing the energy consumption associated with memory access. However, both SRAM and DRAM have their limitations, including high power consumption, lower density, higher cost, refresh overhead, and integration complexity, which may inhibit potential performance enhancements and energy efficiency gains, particularly in energy-constrained applications. Flash memory, despite its many advantages, also has its own set of challenges that could limit its effectiveness in in/near-memory computing. Addressing these limitations will be a crucial step in advancing the development of more efficient in/near-memory computing architectures. This might necessitate exploring alternative memory technologies or innovative design approaches to surmount these hurdles.

### 1.1.3.3 Computer Memory Hierarchy and Addressing the Gap with Emerging Technologies

The evolution of memory technologies over time has led to the development of a diverse array of memory types, each with its own unique set of characteristics. This diversity has given rise to the concept of a memory hierarchy within a computer system, a construct designed to balance the trade-offs between speed, capacity, energy efficiency, and error tolerance. This hierarchy can often be represented as a pyramid. At the top of the pyramid, we find the fastest and most expensive types of memory, namely volatile memories, i.e. static and dynamic random access memories (See Fig. 1.15). At the bottom of the pyramid, we find the slowest but highest-capacity non-volatile memories, which are also the least expensive types of memory: flash memories and magnetic hard drives.

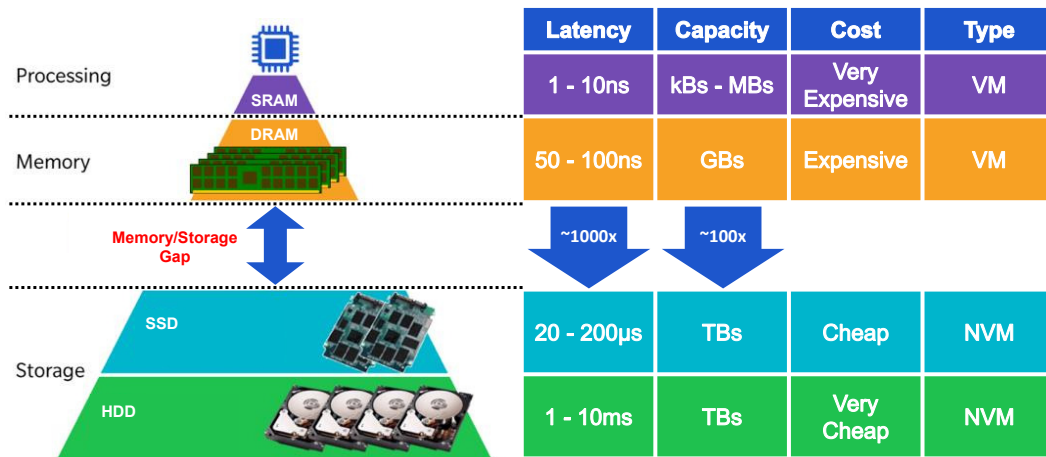


Figure 1.15: **Memory Hierarchy and Addressing the Gap with Emerging Technologies.**

This division within the memory hierarchy gives rise to a distinct gap between volatile and non-volatile memories. This gap elucidates the limitations of existing mature memory technologies in meeting the demands of emerging computing paradigms, such as in/near memory computing, which prioritize energy efficiency [171]. Each of these memory technologies possesses both a trump card feature and an Achilles heel feature. Emerging memory technologies like memristors hold the potential to address this gap in the memory hierarchy by offering a more balanced combination of speed, density, energy efficiency, and non-volatility. They can facilitate the development of more efficient computing architectures and unlock new possibilities for in/near-memory computing, as well as other advanced computing paradigms.

### 1.1.3.4 Emerging Memory Technologies

The goal of many emerging memory technologies is to combine the speed of RAM with the non-volatility data storage, hence their classification as non-volatile RAMs. These devices include memristors, usually called resistive RAM (ReRAM) in the industry, magnetic RAM (MRAM),

phase-change memory (PCM), ferroelectric RAM (FeRAM), and ferroelectric FET (FeFET). These technologies exhibit non-volatility and provide several advantages over traditional nonvolatile memory counterparts, including faster switching speeds, and reduced energy consumption [172]. Each of these devices operates based on distinct physical mechanisms:

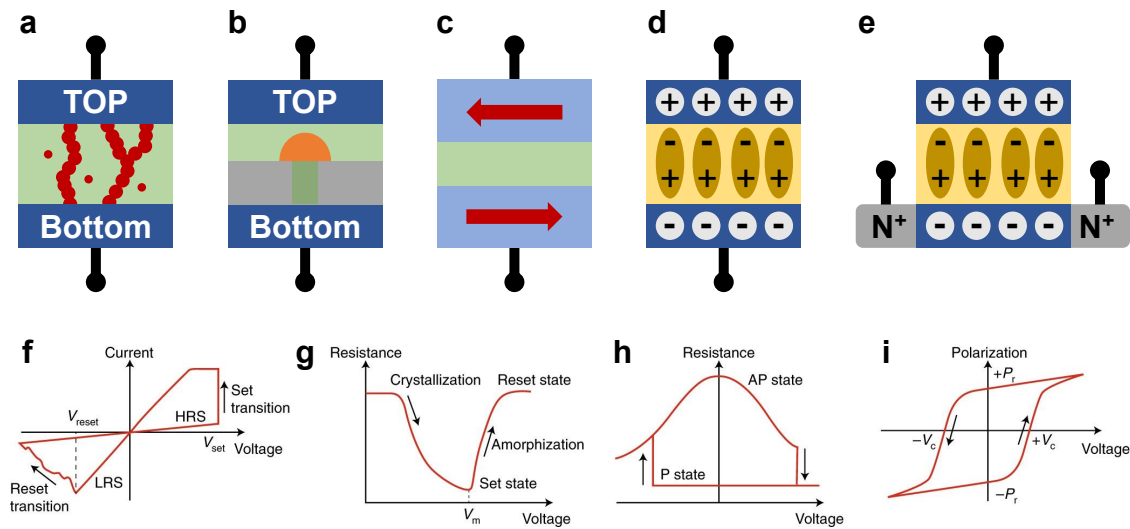


Figure 1.16: **Emerging Memory Technologies.** **a** Resistive RAM structure and its **f** current-voltage (I-V) characteristics for a bipolar switching device. **b** Phase-change memory structure and its **g** resistance change characteristics. **c** Magnetic RAM structure and its **h** resistance-voltage characteristics. **d** Ferroelectric RAM structure and **e** Ferroelectric FET structure and their **i** polarization-voltage hysteretic characteristic. (Reproduced from [11])

**Memristor or ReRAM:** The memristor, a theoretical circuit element proposed by Leon Chua in 1971 [173], was not realized until a disputed claim by HP Labs in 2008 [174]. Despite the controversy, the concept of resistive memory, or ReRAM, sharing similar traits with the proposed memristor, has gained attention [175]. This thesis will use “memristor” and “resistive RAM” interchangeably to refer to these devices. Memristor stores data by modulating the resistance of a dielectric material sandwiched between two metal electrodes (See Fig. 1.16a). The application of an electric field induces the formation or dissolution of conductive filaments within the dielectric material, resulting in a change in the resistance state (See Fig. 1.16f). These resistance states can represent binary data, where high and low resistance states correspond to ‘0’ and ‘1’ respectively, and also can be adapted to store multi-level state data or continuous analog data.

**Magnetic RAM (MRAM):** MRAM stores data using magnetic tunnel junctions (MTJs), which consist of two ferromagnetic layers separated by a thin insulating barrier [176] (See Fig. 1.16c). Data is stored by changing the relative magnetization direction of the ferromagnetic layers. When the magnetization directions are parallel, the MTJ has low resistance, representing a bi-

nary 1. When the magnetization directions are anti-parallel, the MTJ exhibits high resistance, corresponding to a binary 0 (See Fig. 1.16h). MRAM devices can be written and read by applying currents.

**Phase-Change Memory (PCM):** PCM is based on the reversible phase transition of a chalcogenide material, typically a compound of germanium, antimony, and tellurium (GeSbTe or GST) [177] (See Fig. 1.16b). Data is stored by changing the phase of the chalcogenide material between amorphous (high resistance) and crystalline (low resistance) states, representing binary 0 and 1, respectively (See Fig. 1.16g). Phase transitions are induced by applying heat through electrical pulses, which cause the material to melt and subsequently cool rapidly (quench) into the amorphous state or heat and cool slowly to form the crystalline state.

**Ferroelectric RAM (FeRAM):** FeRAM stores data using ferroelectric capacitors, which exhibit spontaneous polarization that can be reversed by applying an electric field [178] (See Fig. 1.16i). Data is represented by the orientation of the polarization, with up and down polarization directions corresponding to binary 0 and 1, respectively (See Fig. 1.16d). The polarization state of a ferroelectric capacitor can be read by applying a voltage and measuring the resulting current, which is proportional to the amount of charge displaced by the polarization reversal. Unlike other emerging memories, the read operation is destructive in these devices.

**Ferroelectric FET (FeFET):** FeFET is a three-terminal device that uses the concept of ferroelectric polarization [179] (See Fig. 1.16e). However, in FeFETs, a ferroelectric material is used as the gate insulator in a Field-Effect Transistor. The data is stored by switching the polarization of the ferroelectric material, which, in turn, modulates the transistor's channel conductivity. Upward and downward polarization directions correspond to binary '0' and '1', respectively (See Fig. 1.16e). The transistor's current state, which represents the stored binary data, can be read by applying a voltage to the gate and measuring the resulting current in the channel. This current is affected by the polarization state of the ferroelectric material and thus indicates whether a '0' or '1' is stored.

In conclusion, the advent of non-volatile RAMs marks a notable shift in memory technology. Each of these technologies possesses distinct operational mechanisms and exhibits a broad array of characteristics. Consequently, they afford a diverse range of potential applications, thereby facilitating the advancement and diversification of memory storage and processing systems.

Memristors and other emerging memories, exhibit a promising range of unique characteristics that are fostering a new landscape of potential applications spanning across various domains. These characteristics encompass parameters such as read and write speed, energy efficiency during read and write operations, endurance, physical area requirements, variability, resolution, susceptibility to read disturbances, cost, and process complexity. Each of these attributes contributes to the versatility of these technologies, thereby rendering them suitable

for a broad spectrum of potential applications [172]. The specific nature of these applications will be inherently determined by the balance of these factors.

**Storage:** Emerging Memories exhibit characteristics that make them a potential candidate for non-volatile storage devices [171]. While their current capacity might not meet the demands of high-storage devices, their potential integration into 3D memory architectures could pave the way for future advancements. For devices with more modest storage requirements, such as IoT devices or microcontroller units, memristors could represent a viable option. Moreover, these emerging technologies might find application in the replacement of DRAM or as third-tier cache memory in low-power, low and medium-speed devices [180].

**Logic:** Emerging Memories can also offer potential utility in the construction of logic gates and circuits [181]. The properties of these devices may enable the development of non-volatile registers and reconfigurable logic gates. It is also conceivable that they could contribute to logic gates requiring fewer components, potentially leading to more compact computing systems.

**Randomness Generation (RNG):** The inherent variability and instability in certain types of emerging memory technologies can serve as a potential source for random number generation [19]. This characteristic could be leveraged in the context of probabilistic computing and cryptographic applications. Additionally, these properties could supply a valuable source of randomness for sampling algorithms [88]. Furthermore, these memory technologies could find use in the creation of physical unclonable functions (PUFs), potentially enhancing hardware security [182].

**In and Near-Memory Computing:** Memristors and other emerging memories, due to their non-volatile memory states, may provide viable utility across a range of computing paradigms, including neuromorphic, digital, analog, and probabilistic computing [168]. They could notably enhance low-power, high-speed vector and matrix multiplication tasks, potentially offering more energy-efficient and high-performance AI inference capabilities at the edge. The capacity of memristors to facilitate in-situ AI learning algorithms presents a compelling approach for AI training at the edge, particularly within the framework of local learning rule-based or brain-inspired algorithms. The prospect of continuous training at the edge might promote greater confidence in AI deployment and proliferation.

### 1.1.3.5 Memristors-Based Artificial Neural Network accelerators

Memristors offer a unique set of attributes that render them particularly suitable for energy-efficient AI systems [103, 172]. What sets them apart is their ability to maintain multiple resistance states. This trait enables them to conduct complex operations and, crucially, to emulate functions of the human brain, an essential component in the development of neuromorphic, or

brain-like, computing. Memristors' compact size and possibility for high storage resolution allow for efficient handling of large volumes of data, a characteristic demanded by AI applications and a distinct advantage over solutions such as SRAM. The potential for 3D configuration could further augment their data storage capacity, answering the escalating demand for memory in AI applications. Their efficiency is bolstered by fast, low-energy write and read operations, which offer both speed and energy benefits. Furthermore, the simplicity of their structure and fabrication process could lead to cost savings. The compatibility of memristors with the CMOS back-end-of-line process facilitates their integration into existing manufacturing workflows.

Despite challenges such as consistency and reliability, the growing body of research and the increasing application of memristors in AI hardware systems underscore their potential. Until now most research has focused on implementing artificial neural networks (ANNs).

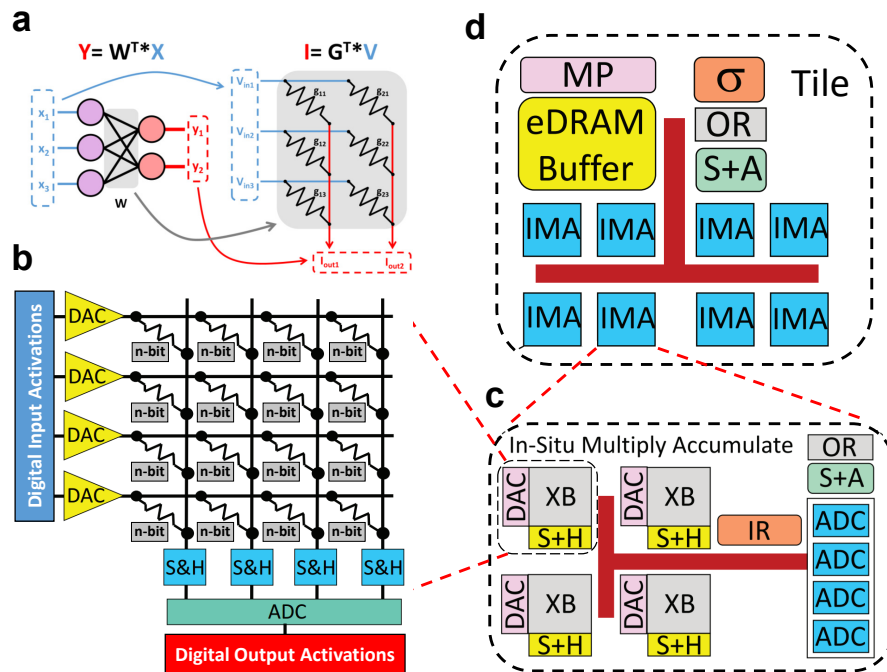


Figure 1.17: **In-Memory computing with Memristor crossbars for artificial neural network. a** Neural network with three inputs and two outputs mapped on a memristor crossbar of three rows and two columns. The multiply-and-accumulation operation can be performed in the analog regime, taking advantage of Ohm's Law and Kirchhoff's Law. **b** A memristor crossbar used as a vector-matrix multiplier, including ADCs, DACs, and digital input and output circuitry. This crossbar is a main element in the ISAAC architecture hierarchy, used to build **c** the In-Situ Multiply-and-Accumulate block that is part of **d** the ISSAC Tile block. (Reproduced from [12])

**Memristor crossbar for In-Memory Computing of Artificial Neural Network:** A key operation in neural networks is the multiplication of weights and input data followed by an accumulation of the products, commonly referred to as multiply-and-accumulate. Memristor

crossbar arrays can be employed to perform this operation in an analog, in-memory fashion, taking advantage of Ohm's Law and Kirchhoff's Law. In such arrays, the conductance of the memristor devices represents the synaptic weights of the neural network. In a weight-stationary dataflow scheme, input voltages, representing input activations, are applied to the rows, and the resulting currents in the columns correspond to the dot products, i.e., the neuron outputs (See Fig. 1.17a). Numerous realizations of this proposal have been demonstrated recently [103, 168, 183–186]. By performing multiply-and-accumulate operation directly within the memory array, memristor-based in-memory analog computation can significantly reduce data movement between memory and processing units, which is a major source of energy consumption in traditional architectures. Additionally, the parallelism inherent in the crossbar structure enables a high degree of concurrency, potentially leading to substantial improvements in computational throughput. Furthermore, memristor-based analog computing can leverage the continuous conductance levels of memristor devices to represent multi-bit synaptic weights, enabling possibility for higher precision calculations compared to binary-weighted neural networks. This can lead to better accuracy in inference tasks without incurring the energy and area overhead typically associated with higher precision digital computation.

However, a practical implementation of memristor crossbars in industrial products needs a deep study of all needed parts for a fully reliable system, not only relying on conceptual potentials. Implementing an ANN accelerator with in-memory analog computing with a memristor crossbar is still a challenging research subject. This is due to device imperfections that present a substantial challenge [88, 168, 172]. Achieving precise control over each memristor's resistance state, which signifies a synaptic weight in neural networks, is still limited due to the current memristor devices variability. This lack of precision can lead to limited reliable device resolution, heavy precise write periphery circuits, and affects on computational accuracy [25]. From a circuit level, higher design complexity, as MAC operations are done in analog and other operations are in digital, necessitates the conversion between analog and digital signals, therefore, complex Analog to Digital Converters (ADC) and Digital to Analog Converters (DAC) circuitries are required, with high-precision, high-speed and tolerant devices noise (See Fig. 1.17b). From a system level, as the weights are stationary, a careful design consideration needs to be taken for the dimensions of memristor crossbars and the capacity of volatile memory for activations: this is for giving more degree of freedom for mapping, applying pipelined computation, reconfiguration of the system for different neural networks models, and to increase data usage and reuse for a purpose of increasing the efficiency of the system (See Fig. 1.17c-d). Addressing these challenges necessitates advancements in memristor device technology, to improve device characteristics, circuit design techniques, improve read write circuitry and techniques, system-level architectures, and algorithmic level, such as developing hardware friendly AI models, based on network pruning, network, reduced precision models.

A famous example, to be mentioned here is the ISAAC (In-Situ Analog Arithmetic in Crossbars) accelerator [12], which has been proposed for convolutional neural networks (CNN) ac-

celerators. This architecture leverages memristor crossbar arrays at its core to store synaptic weights and perform dot-product computations (Fig. 1.17b). The system's core is built around memristor crossbar arrays integrated within each of its multiple nodes or tiles (See Fig. 1.17c-d). The system tackles several of the challenges mentioned above, by limiting device resolution to 2-bits and using sequential inputs to reduce analog-to-digital converter (ADC) and digital-to-analog converter (DAC) overheads. Furthermore, through a meticulous design space exploration, ISAAC studied the optimal balance of chip area dedication to storage, computation, buffers, and ADCs, resulting in a substantial boost in throughput, energy efficiency, and computational density for comprehensive CNN and Deep Neural Network (DNN) applications. This work addresses the challenges associated with adopting a solution based on emerging approaches that encompass interdisciplinary fields.

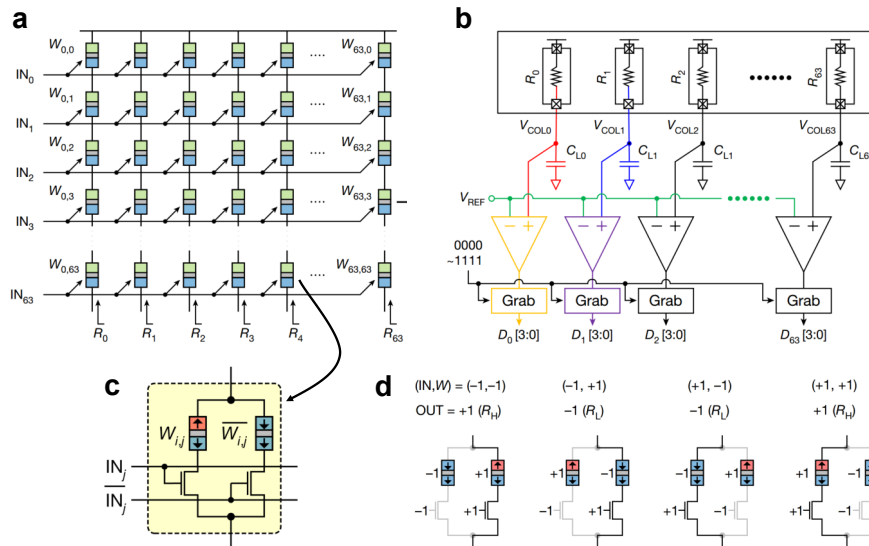


Figure 1.18: **In-Memory computing with resistance summation for artificial neural network.** **a** Resistance summation crossbar array architecture. **b** Time-domain readout method. A lumped capacitor and distributed parasitic capacitors in the array are charged, and the time taken for the voltage at the end of the column to reach a reference voltage is measured, correlating to the column resistance. The resistance value represents the dot product of the input vector and the weight vector. **c** Bit-cell structure, which combines two parallel paths, each comprising a resistive device and a MOSFET in series. **d** Implementation of an analog XNOR operation, the multiply operation for the binary neural network, by the bit-cell structure. (Reproduced from [13]).

**A different approach for Memristor crossbar.** In-memory computing using memristor crossbars based on current summation faces dimensional constraints. This arises from the high current flow associated with numerous parallel resistors, particularly in the case of low-resistance states. This high current can lead to an increased peripheral circuitry size and enhanced impact of parasitic resistors in routing metals. Addressing this, researchers from the Samsung



Advanced Institute of Technology have proposed a novel crossbar array architecture that leverages resistance summation for analogue multiply-accumulate operations, instead of current summation [13] (See Fig. 1.18a). The architecture is built on an MRAM crossbar array, chosen for its low resistance states. As MRAM is confined to only two states, this architecture has been utilized to implement a binary neural network model for image classification tasks. The architecture's bit-cell combines two parallel paths, each comprising an MRAM tunnel junction (MTJ) and a MOSFET in series (See Fig. 1.18c). In this configuration, one path's FET gate is driven by a binary input voltage, while the other is driven by the complement of the input voltage. Each path stores a synaptic resistance weight, with one path holding the weight and the other its complement. The chosen path, determined by the input voltage, results in the bit-cell output, thus implementing an analog XNOR operation—the multiply operation for the binary neural network (see Fig. 1.18d). The bit-cell output resistances are connected in series in each column of the array and their sum yields the column resistance. This process replaces the current sum in conventional crossbar arrays, with the column resistance being the dot product of the input vector and the weight vector. The column resistance is read via a time-domain method; a lumped capacitor and distributed parasitic capacitors in the array are charged, and the time taken for the voltage at the end of the column to reach a reference voltage is measured, correlating to the column resistance (see Fig. 1.18b). This time delay is captured by a time-to-digital converter (TDC) that extracts the resistance.

Despite MRAM's low resistances, this architecture promises lower power consumption, as computations are based on charge and discharge currents rather than steady summed currents. A limitation, however, is the binary nature of the MRAM, which may necessitate an increased network size or longer computing time to achieve desired performance. An effective countermeasure is the integration of memristors, which, unlike binary MRAM, can store a broader range of resistance states, enhancing the precision of stored weight values and promoting more accurate computations within the neural network model.

## 1.2 The Concept of Near-Memory Compute Architecture for a Bayesian Machine

As delineated in the introduction of this thesis, positioning AI at the edge could mitigate certain trust concerns associated with artificial intelligence. Bayesian reasoning, or Bayesian inference, is an artificial intelligence approach that could better adapt than neural networks to safety-critical applications, where explainable decisions with uncertainty-quantification are required [82, 187]. Bayesian reasoning is a probabilistic framework that permits decision-making in situations with incomplete information, maximally incorporating all available evidence, assumptions, and prior knowledge [188, 189]. Within this approach, reasoning is fully explainable and excels at ‘small data’ situations, as it is able to incorporate prior expert knowledge [190]. It can also estimate the certainty of its prediction [187], which is a challenge for neural networks. Bayesian models are not directly brain-inspired but have been connected to biological intelligence [191–196].

However, although Bayesian reasoning requires considerable memory access, implementing it near-memory is more challenging than for neural networks. In a Bayesian approach, networks feature a topological nature, but in a way that is more subtle than neural networks. Bayesian reasoning is usually implemented on conventional computers [197], microcontroller units [198, 199], or graphics processing units [200]. Several works have also implemented it on large field-programmable gate arrays [201–204], and CMOS-based application-specific integrated circuits [205]. However, the energy efficiency of such approaches is always limited by the cost of memory access to the external dynamic random-access memory.

Because Bayesian inference does not use multiply-and-accumulate operations, strategies commonly used in neural networks accelerators, such as relying on analogue computation (see section 1.1.3.5), do not bring the same benefit and would have a very high cost in terms of periphery circuit. Therefore, we chose to use a strategy in which memristors are used in a binary fashion and read by tiny, robust and highly energy-efficient sense amplifiers. As a result, we developed hardware systems that we call Bayesian machines. Based on memristors and near-memory computing, those machines offer features not commonly seen in neural network accelerators: they do not need any calibration process; they are robust to device imperfection without the need for any compensation circuitry; and they can function over a broad range of voltage without any adjustment, making them particularly useful in environments with unreliable power supply, such as those based on energy harvesting.

The foundation of our Bayesian system is Bayes’ theorem, which is a fundamental concept in Bayesian statistics that allows for the updating of probabilities based on new observed evidence:

$$P(Y|O) = \frac{P(O|Y)P(Y)}{P(O)}. \quad (1.1)$$

The theorem states that the posterior probability of a hypothesis or event given some observed evidence  $P(Y|O)$  is proportional to the product of the likelihood of the observed evidence given the hypothesis  $P(O|Y)$  and the prior probability of the hypothesis  $P(Y)$ . Since calculating the probability of evidence  $P(O)$  can be challenging, and this factor only acts as a uniform multiplicative coefficient, the product of the prior and likelihood is often considered as a simplified approach to computation:

$$P(Y|O) \propto P(O|Y)P(Y). \quad (1.2)$$

The probabilities updating process enables us to revise our beliefs about a hypothesis as we receive new evidence. Interestingly, this updating process mirrors how humans learn and understand new concepts. When humans encounter new information, they update their beliefs to reflect this new knowledge. This process of updating beliefs based on new information is a crucial aspect of the learning process and is analogous to the Bayesian updating of probabilities using Bayes' theorem.

Bayesian programming is a complex and broad discipline that cannot be fully summarized in a subsection of a chapter. For a comprehensive understanding, we recommend reading the "Probabilistic Graphical Models"[206] and "Bayesian Programming" [189] books. In this section, we focus on the adaptation of Bayesian inference models to a near-memory hardware architecture that we call the Bayesian machine architecture.

Let us take an example where we try to evaluate the probability of an event  $Y$ , e.g., the occurrence of a medical emergency, based on a collection of observations. A value of one for  $Y$  might represent the occurrence of a minor stroke, a value of two a major stroke, and a value of zero the absence of any stroke. Bayes' law then provides the probability that these situations are currently occurring, based on their probability to happen at any time (the prior  $p(Y = y)$ ) and likelihood factors  $p(O_1, O_2, \dots, O_n|Y = y)$  that model the behavior of the sensors  $O_1, O_2, \dots, O_n$  in the absence or presence of a stroke:

$$p(Y = y|O_1, O_2, \dots, O_n) \propto p(O_1, O_2, \dots, O_n|Y = y) \times p(Y = y). \quad (1.3)$$

The likelihood factors feature a prohibitive memory cost, which grows exponentially with the number of observations  $n$ . In real-life settings, this cost can be alleviated. In our example, sensors that measure distinct aspects of the patients (e.g., heart rate and body temperature) may often be considered conditionally independent (meaning that once given the knowledge that a stroke is currently happening, the heart rate and body temperature values can be regarded as statistically independent processes). If all observations are conditionally independent, equation 1.3 simplifies to

$$p(Y = y|O_1, O_2, \dots, O_n) \propto p(O_1|Y = y) \times p(O_2|Y = y) \times p(O_3|Y = y) \dots \times p(O_n|Y = y) \times p(Y = y), \quad (1.4)$$

with a memory cost for the likelihood now growing linearly with  $n$ , and likelihood factors now becoming easy to model based on measurements of the sensor values, e.g., during the occurrence or in the absence of a stroke.

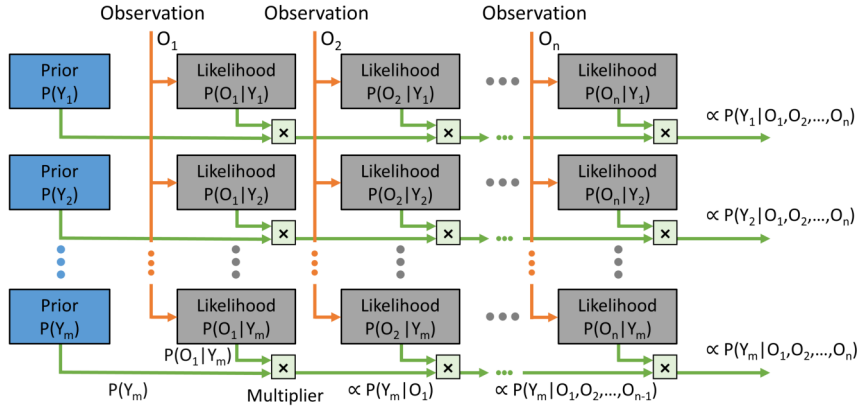


Figure 1.19: **General architecture of the Bayesian machine.** The likelihoods are stored in likelihood memory arrays implemented by memristor arrays. Observations from the real world choose the appropriate probability values from likelihood memory arrays, based on which the probability values are read from likelihood arrays, which are multiplied by multipliers. At the output, the generated results encode the posterior distribution.

Our memristor-based Bayesian machine concept, presented in Fig. 1.19, implements equations such as eq. 1.4 in a topological manner. Each likelihood factor is implemented using independent memory arrays, and multiplications are performed physically close to these memory arrays. The multiplication result is then passed to the next memory array. The observations  $O_1, \dots, O_n$  effectively act as addresses for the memory arrays, telling which likelihood values should be read.

The architecture in Fig. 1.19 presented a case where all observations can be considered conditionally independent, which is not always the case. In particular, When two redundant sensors measure the same phenomenon, they may not be regarded as independent in a good model, even conditionally to the inferred variable. For example, if observations  $O_2$  and  $O_3$  are not conditionally-independent, the  $p(O_2, O_3 | Y = y)$  likelihood may not be factorized, and an appropriate Bayesian inference model may read such as eq. 1.5

$$p(Y = y | O_1, O_2, \dots, O_n) \propto p(O_1 | Y = y) \times p(O_2, O_3 | Y = y) \times \dots \times p(O_n | Y = y) \times p(Y = y). \quad (1.5)$$

This model can still be implemented by a memristor-based Bayesian machine. In that case,

observations  $O_2$  and  $O_3$  should be pooled into a single column, and their joint value should be used to address the likelihood arrays of this column, as presented in Fig. 1.20. Nevertheless, it should be highlighted that the memory cost of the Bayesian machine grows with the number of non-conditionally independent variables.

It should also be noted that in the case of a uniform prior, the prior blocks of the Bayesian machine may be removed entirely.

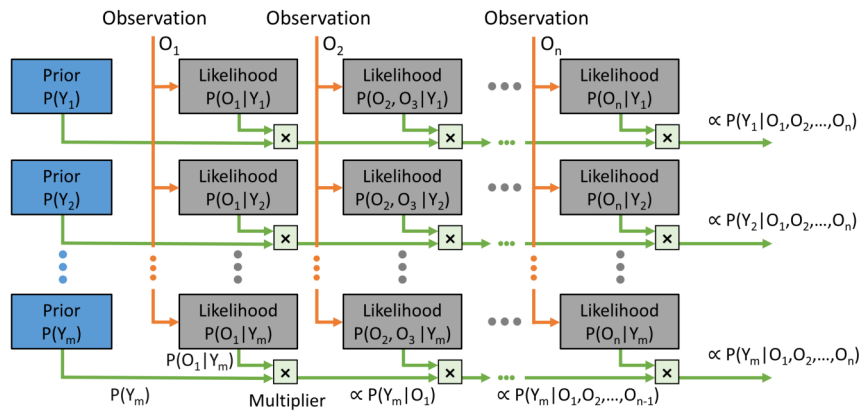


Figure 1.20: **Architecture of the Bayesian machine with non-conditionally independent observations.** This architecture performs non-naive Bayesian inference following eq. 1.5, by pooling observations  $O_2$  and  $O_3$  into the same column.

An important challenge of the memristor-based Bayesian machine is that multiplications are normally an area-expensive operation in CMOS, raising a concern if a multiplier is associated with each likelihood memory array. For this reason, we adopted two promising computing approaches to implement several Bayesian machines with different computing styles:

- Stochastic computing [207, 208] (see Chapter 3): a computing paradigm encodes probabilities as streams of random bits, where, at each clock cycle, the probability for the bit to be one is just the encoded probability. The multiplication of probabilities can then be achieved using simple AND gates, with an extremely minimal area cost [207].
- Logarithmic computing [209, 210] (see Chapter 4): a computing paradigm that encodes probabilities in the logarithmic domain. The multiplication of probabilities can then be achieved using simple addition operation.

An important aspect shared by both these models of computation is that the memristor-based Bayesian machine reduces data movement considerably. Due to its simplicity, the Bayesian machine just looks like a memory chip – we call it a “natively intelligent” memory.

## 1.3 Steps of the Bayesian Machine projects

Making a chip, whether in an industrial company or in a research lab, is a complex and demanding process that requires a combination of technical expertise, advanced manufacturing techniques, and significant resources. For industry, the need for advanced engineering, sophisticated manufacturing techniques, and significant investment has led to a limited supply chain dominated by a few companies at the forefront of cutting-edge technology. Meanwhile, research labs often face greater challenges due to limited resources, including funding, personnel, and equipment, making it difficult for them to access the necessary advanced tools and techniques to produce high-quality chips.

The focus on exploring new ideas and technologies in research labs often involves taking risks and experimenting with novel approaches, such as new hardware architectures, devices, and computing paradigms, which can increase the risk of having errors or defects on the fabricated chip and make the design and verification process more complex using less mature Electronic Design Automation tools. Thus, the challenges faced in projects involving chip fabrication highlight the importance of a productive multidisciplinary collaboration. Our projects involving the Bayesian machines are the result of the successful collaboration of a group of researchers from four research entities: Centre for Nanosciences and Nanotechnologies (C2N) in the Paris-Saclay territory, Institut Matériaux Microélectronique Nanosciences de Provence (IM2NP) in Marseille, CEA-Leti in Grenoble, Institute of Intelligent Systems and Robotics (ISIR) in Paris, and a start-up company (HawAI.tech) based in Grenoble.



Figure 1.21: **Affiliation of collaborating research entities in the Bayesian machine project.** Including C2N, IM2NP, CEA-Leti, ISIR, and HawAI.tech. Along with image of our paper on the cover of Nature Electronics Journal [14].

The aim of a Bayesian machine project is to develop a highly optimized and efficient Bayesian machine by incorporating memristors into an integrated circuit. To achieve this, a collaborative approach is taken, with experts in Bayesian theory, memristor device modeling and characterization, and integrated circuit design working together. The project consists of several main steps.

The theory and early model of the Bayesian machine were developed by Tifenn Hirtzlin (then at C2N) under the supervision of Damien Querlioz (C2N), in collaboration with Bayesian theory experts Jacques Droulez and Pierre Bessiere (ISIR). Then, an intensive study of design choices and computing paradigms was performed in collaboration with Marc Bocquet (IM2NP), who specializes in memristor device modeling and characterization, and Jean-Michel Portal (IM2NP), an expert in integrated circuit design. I did the actual design of the first Bayesian machine "The stochastic Bayesian machine", in collaboration with Tifenn Hirtzlin, and of the second Bayesian machine "The logarithmic Bayesian machine", in collaboration with Clement Turck (C2N), all under the supervision of Jean-Michel Portal and Damien Querlioz.

After the design was finalized, the chip was sent to a foundry for fabrication, with Elisa Vianello (CEA-Leti) leading the process. The fabrication process is based on a hybrid CMOS/memristor technology, which goes through two phases. The first phase involves the fabrication of the CMOS part using a conventional silicon-based process with a 130-nanometer commercial technology. The second phase involves the integration of memristor devices using an emerging technology process from the CEA-Leti research center.

Once the chips are received, the electrical characterization of the stochastic machine was performed by Marc Bocquet and Tifenn Hirtzlin; the electrical characterization of the logarithmic machine was performed by Clement Turck and I, under the supervision of Marc Bocquet and Damien Querlioz. Both systems were tested using a custom-designed measurement setup, and in parallel, we designed scaled-up versions of the Bayesian systems with Clement Turck to implement applications based on Bayesian inference models, adapted to the Bayesian machines.

Using the scaled-up system and a homemade energy analysis framework developed by Clement Turck and I, an energy versus performance study is conducted, and the results are benchmarked with conventional computing units. Finally, the performance of the Bayesian machine is evaluated in real-world application, a hand gesture recognition task, the task was developed by Raphael Laurent (HawAI.tech). The potential benefits and limitations of the Bayesian machine are evaluated. Our works were rewarded by one published article in the Nature Electronics journal [14], and one article presented at the DATE 2023 conference [86].

### 1.3.1 Design Flow for Memristor-based Chips

The Bayesian machine is an application-specific integrated circuit (ASIC), i.e., an integrated circuit designed for a particular use and. To design these circuits, a complex design flow needs to be respected. A conventional design flow of a chip using mature technologies (available

for industries) is a multi-step process that starts from a concept or idea to the creation of a Graphic Design System (GDS) mask of a functional integrated circuit ready to be sent to fabrication. In this multi-step process, several software tools provided by EDA companies such as Cadence and Synopsys are used. Technology libraries are needed, provided by the design kit of the foundry.

As shown in Fig. 1.22, the design flow begins with system specifications, where the definition of concept and requirements for the chip is defined, including the intended use case, performance requirements, and power constraints. The next step is the architectural design, where the overall architecture of the chip is determined, including the number and types of components, the connections and the data flow between them. The digital logic of the chip is then designed using a high-level hardware description language in the RTL (register-transfer level) design stage. This is followed by the logic synthesis step, where the RTL design is transformed into a gate-level representation. The physical design step involves laying out the components of the chip and defining the interconnections between components. Verification is an important part of the design flow, where the chip design is tested and validated at various stages to ensure that it meets the specifications. The final step is generating the GDS file to be sent for fabrication, where the chip is manufactured using specialized techniques such as photolithography.

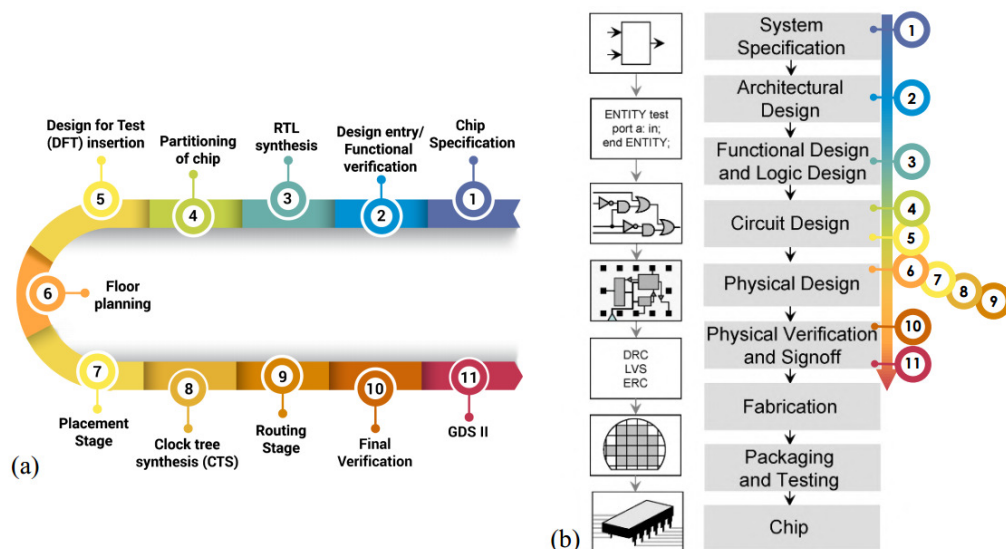


Figure 1.22: **Overview of an Integrated Circuit Design Flow.** **a** Computer-aided steps performed with EDA Tools. At the end, GDS mask layouts are obtained, ready for the fabrication process (Reproduced from [15]). **b** Diagram of the main steps for making a chip from system specification to ready-to-use chip (Reproduced from wikipedia).

The memristor-based Bayesian machines are hybrid CMOS/memristor integrated circuits that embed memory arrays within logic. Due to the lack of a foundry design kit that supports



such designs, and the need to use multi-supply voltages, we developed a semi-automated design flow for our first Bayesian machine (the stochastic Bayesian machine). The memristor arrays and their mixed-signal peripheral circuitry were manually designed, placed, and routed using the Cadence Virtuoso electronic design automation (EDA) tool. To enable the exploration of various programming regimes for the memristors, we designed the programming circuitry with wide transistors and large safety margins. Analog simulations were performed using the Siemens Eldo and Cadence Spectre simulators.

On the other hand, we described all digital computation blocks using the SystemVerilog hardware description language and verified their logical correctness using the Cadence NC-Verilog Simulator. The digital circuits were synthesized using either the Cadence Encounter RTL Compiler (stochastic Bayesian machine) or Cadence Genus Synthesis Solutions (logarithmic Bayesian machine), and then placed and routed using the Cadence Encounter RTL-to-GDSII tool (stochastic Bayesian machine) or the Cadence Innovus system (logarithmic Bayesian machine). The digital circuits employed thin gate oxide high-threshold transistors.

In the case of the stochastic Bayesian machine, the resulting computation blocks' layouts, as well as those of memory blocks, were manually placed and routed in a full-custom fashion. We performed design rule checks, layout-versus-schematic comparison, and antenna effects design rule checks using dedicated Calibre EDA tools to ensure the final design's physical verifications.

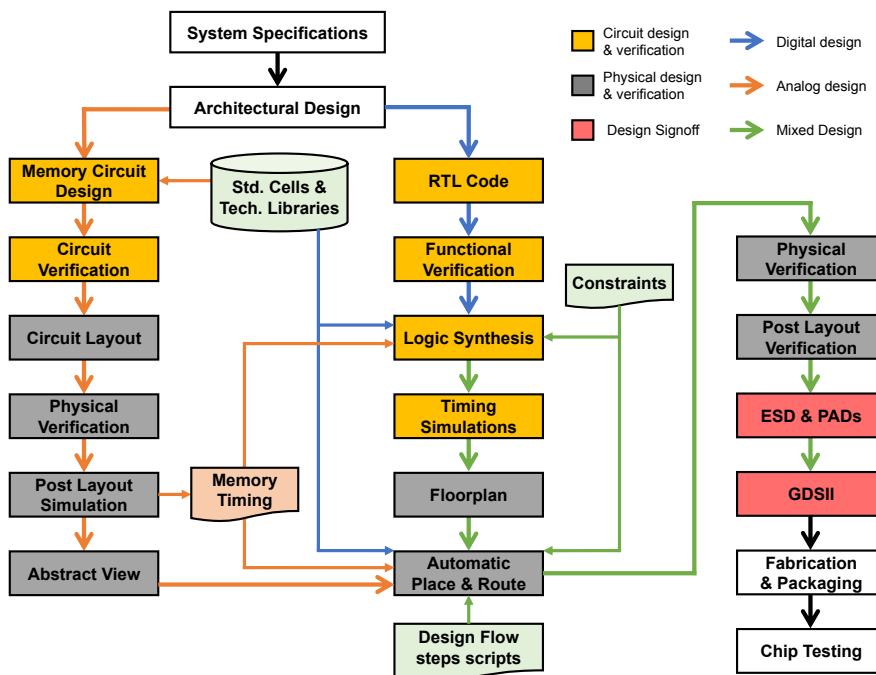


Figure 1.23: Diagram illustrating our custom-developed automated design flow for integrating mixed digital and memory circuits.

To expedite the physical design process for the Logarithmic Bayesian machine, the next generation of Bayesian machines, we developed a fully-automated place and route flow, shown in Fig. 1.23. This flow automatically places and routes the memory blocks, which consist of a memristor array with its mixed-signal peripheral circuitry, alongside digital logic standard cells. This significantly reduces the time and effort required for the design process. Prior to the synthesis step, we developed a Liberty Timing File for the memory cell, which represents the timing and power parameters associated with the cell. In addition, we generated an abstract view of the memory cell, which is necessary for automatic layout tools. We then only need to do some modifications to the standard automated place and route scripts and use a custom-designed floor plan. With these changes, we are able to complete the physical design process of any digital hybrid CMOS/memristor-based chip, from the placement stage (step 7 in Fig. 1.22) to the generation of a GDSII file (step 11 in Fig. 1.22) using the Cadence Innovus Implementation System. We then perform further physical verifications of the final design, including design rule checks, layout-versus-schematic comparison, and antenna effects design rule checks, using dedicated Calibre EDA tools. By implementing this automated flow, we were able to reduce the time required for the physical design process from over a month to less than two days, which significantly sped up our research and development efforts.

Once the design was finalized, the chip was sent to fabrication. The CMOS part of our test chip was fabricated using a low-power foundry 130-nanometer process with four layers of metals. The memristors were fabricated on top of exposed vias and composed of a TiN/HfO<sub>x</sub>/Ti/TiN stack. The active HfO<sub>x</sub> layer was deposited using atomic layer deposition and is 10-nanometers thick. The Ti layer was also 10-nanometers thick, and the memristor structure had a diameter of 300 nanometers. Finally, a fifth layer of metal was deposited on top of the memristors.

### 1.3.2 Measurement Setup for Memristor-based Chips

When preparing a measurement setup for a memristor-based die, the choice of packaging is critical. Packaging, which involves enclosing an integrated circuit in a protective case or enclosure, is primarily aimed at safeguarding the fragile silicon chip from mechanical damage, moisture, corrosion, and other environmental factors that could impair or destroy its performance. Additionally, it facilitates the mounting of electrical contacts for connecting the die to the printed circuit board (PCB).

Our hybrid CMOS/memristor chips are fabricated in two phases, including the input/output pads that are primarily intended for a characterization task, without built-in electrostatic discharge (ESD) protection. To overcome this limitation, we designed and implemented our own custom ESD protection circuit, necessitating additional precautions during measurements. However, packaging a chip with only custom ESD protection is a risky decision, so we produced two batches of chips: one non-packaged batch (Fig. 1.24a-b) and one packaged batch (Fig. 1.24e), which required two different measurement setups.

We used a custom-made 25-pads probe card (Fig. 1.24c) and a dedicated printed circuit

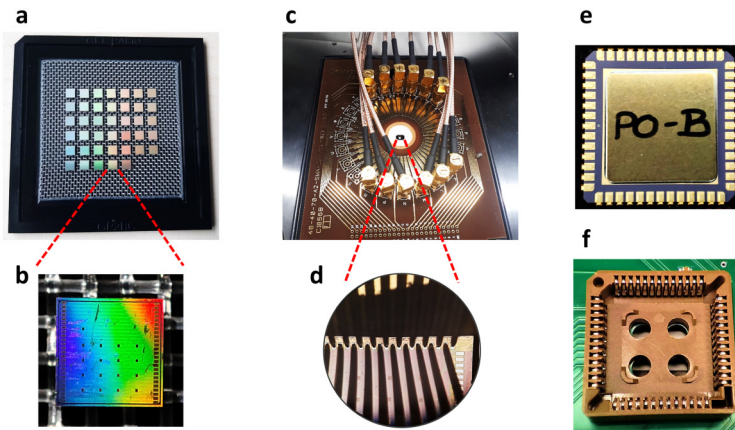


Figure 1.24: **Packaging or probe testing of Bayesian machine dies.** **a** Non-Packaged batch (Logarithmic chip) and **b** a zoom-in on one die. **c** The custom-made 25-pads probe card, used within the probe station to connect the pads of the non-packaged dies to SMA connectors. **d** The operation of connecting the 25 micro-probes to the 25 chip pads. **e** Packaged die with a J-Lead Ceramic Chip Carrier of 52 pins (JLCC52) and **f** a plastic leaded chip carriers (PLCC) sockets, a chip carrier used to form connections between packaged chips and PCB. Chips can be easily exchanged or removed.

board (PCB) with level shifters and other discrete elements connected via SubMiniature A (SMA) connectors (Fig. 1.25b) to probe test the non-packaged dies. The PCB connects the inputs and outputs of our test chip to an ST Microelectronics STM32F746ZGT6 microcontroller unit, two Keysight B1530A waveform generator/fast measurement units, and a Tektronix DPO 3014 oscilloscope. While the microcontroller is connected to a computer via a serial connection, other equipment is connected to the computer via a National Instruments GPIB connection.

For packaged dies, we designed a custom PCB that differs from the one used for non-packaged dies (Fig. 1.25a). The packaged dies can be easily plugged and unplugged into a package adapter (socket) soldered on the PCB. The inputs and outputs of this PCB are connected to the same hardware and use the same software as the non-packaged die setup. For both measurement setups, the tests are conducted using Python within a single Jupyter notebook that controls the entire setup (Fig. 1.25).

Before using the Bayesian machine, the memristors must undergo a unique “forming” operation to create conductive filaments. This is done memristor-by-memristor. Once formed, the memristors can be programmed in low-resistance or high-resistance states (LRS or HRS). We need to conduct characterization experiments to explore the digital programming conditions and identify the appropriate programming voltages. Once we determined the correct programming voltages, we can program the likelihoods in the memristor arrays. High programming voltages are no longer necessary once the likelihoods are programmed, and the test chip can be used in its normal mode to perform Bayesian inference based on observations (more details about chip testing will be reported in the next chapter).

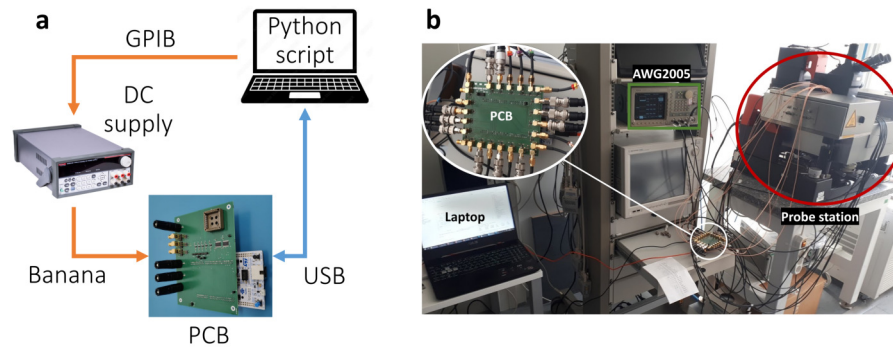


Figure 1.25: **Measurement setups for Bayesian machine systems.** **a** Setup for the packaged dies. **b** Setup for the non-packaged dies.

### 1.3.3 Task Implementation and Energy analysis

The characterization and testing of the Bayesian chip were essential components of our research project. After successfully verifying programming and inference, it was crucial to conduct an energy and performance analysis of the system. To achieve this, we worked on a real-world task, the hand gesture recognition task, based on inputs measured by an inertial measurement unit (IMU). The ultimate objective of the system was to accurately identify the hand gestures performed by a user wearing the IMU. All details about this part of the work are provided in the next chapter.

Fig. 1.26 recapitulates the different steps of the hand gesture recognition implementation on the Bayesian machine, from training to on-chip operation:

- **Collect training data.** Any project starts by collecting training data.
- **Implement the likelihood model.** The likelihoods of the Bayesian models are computed. In our sample gesture recognition task, likelihoods are modeled by fitting Gaussian laws on the training data. In other situations, likelihood models may also be obtained based on expert knowledge or prior information [189].
- **Normalize and quantize likelihoods.** For improving the efficiency of stochastic computing, likelihoods are normalized per column by the maximum likelihood value of the column. Likelihoods are then discretized as eight-bit integers, and models are quantized to the number of observation values supported by the Bayesian machine.
- **Program the Bayesian machine.** Likelihood values are programmed to the memristors of the Bayesian machine following the methodology described in the next chapter.
- **Use the Bayesian machine to do inference.** The Bayesian machine can finally be used to infer variables based on observations, using the methodology described in the next chapter.

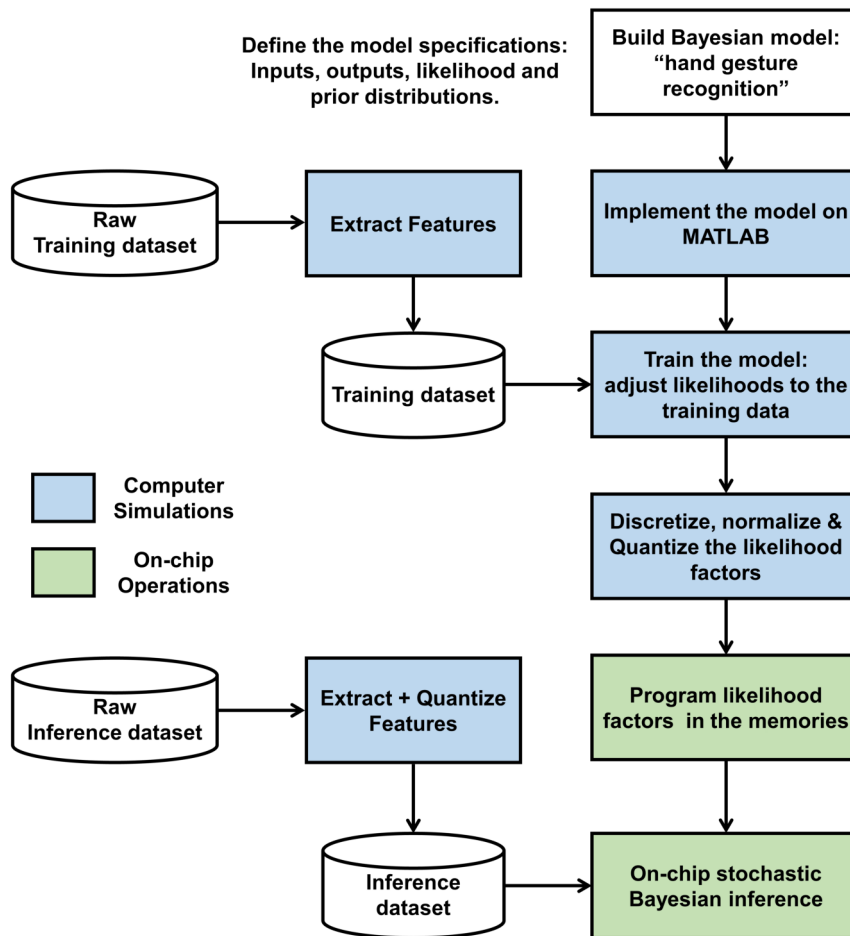


Figure 1.26: **The different steps of a project with the Bayesian machine, from training to on-chip inference.** a Diagram summarizing the main steps of a project involving the Bayesian machine, from Bayesian model building to using the Bayesian machine to perform on-chip inference.

## 1.4 Comparison of our Bayesian Machines with Other Nanotechnology-Based Machine Learning Accelerators

### 1.4.1 Our Bayesian Machines vs. Nanodevice-Based Neural Networks

We saw in section 1.1.3.5 that in recent years, there have been significant advancements in the use of memristors and emerging resistive memories such as phase-change memory (PCM) and spin-torque magnetic random access memory (MRAM) for machine learning accelerators. These implementations aim to closely associate memory and computing functions to eliminate the energy cost of the von Neumann bottleneck. While these approaches differ in their use of memory devices, they all show great potential for improving machine learning performance.

Since 2020, several high-profile machine learning accelerators that employ in- or near-memory computing based on emerging memories have been published. Table 1.1 provides an overview of some of these accelerators, as well as our own Bayesian machine accelerators. While these published works target neural network implementation, our work is focused on the first fully fabricated memristor-based Bayesian machine. Unlike neural networks, Bayesian computing requires higher precision for storing likelihood, and does not require multiply-and-accumulate (MAC) operations, which limits the benefits of analog computation in neural networks.

The memristor-based neural network accelerators have reached better technological maturity than memristor-based Bayesian system, with several systems implemented in sub-30-nm CMOS [13, 186, 211]. The memory array in [186], based on a fully commercial technology, possesses an impressive number of 2M devices. At the same time, state-of-the-art memristor-based network accelerators usually feature a single memory array [13, 186, 211–213], or in the case of [214], three independent memory arrays.

Some neural network implementations exploit the analog storage feature of memristive technologies and phase change memories ([214], [212], [213] and [211]). Others rely on single-bit basic cells to simplify periphery circuitry [186] or due to device limitations. Note that [13] relies on single-bit-per-cell storage, but still uses analog computation to compute neuronal activation function (using resistance sum, as we explained in subsubsec:Inmemoryanalog).

Despite the inherent complexity of analog or mixed-signal circuitry, utilizing analog storage can be advantageous for neural network implementations for two primary reasons. Firstly, the fundamental operation of neural networks, namely multiply-and-accumulate (MAC), can be naturally realized through the use of Ohm's law and Kirchhoff's current law when memristors are utilized in this way. As a result, several devices can be read simultaneously during in-memory computing, utilizing the same periphery circuitry, which significantly decreases the energy cost of analog and mixed-signal periphery circuitry through parallelism. Secondly, neu-

ral network inference only requires low precision for synaptic weights, further enhancing the suitability of analog storage for this purpose.

	<b>This work</b>	Yao et al., 2020[212]	Wan et al., 2020[213]
Application	<b>Bayesian inference</b>	Neural Network	Neural Network/RBM
Computations	<b>Bayes' law</b>	MAC	MAC
Device	<b>HfOx memristor</b>	HfOx/TaOx memristor	HfOx/TaOx memristor
CMOS node	<b>130 nm</b>	130 nm	130 nm
Basic cell	<b>2T2R</b>	1T1R	1T1R
Levels per cell	<b>SLC</b>	Analog	Analog
Read circuit	<b>PCSA</b>	Analog + ADC	Analog + CDS
Number of arrays	<b>16 (connected)</b>	1	1
Number of devices	<b>16×8×8</b>	16 × 128	256 × 256
Inference voltage	<b>0.62-1.2V</b>	Predet.	Predet.
Need for calibration	<b>No</b>	ND	Yes

	Xue et al., 2021[186]	Jung et al., 2022[13]	Khaddam et al., 2022[211]
Application	Neural Network	Neural Network	Neural Network
Computations	MAC	MAC	MAC
Device	Proprietary RRAM	MRAM	PCM
CMOS node	22 nm	28 nm	14 nm
Basic cell	1T1R	2T2R	8T4R
Levels per cell	SLC	SLC	Analog
Read circuit	Sense amplifier	Analog + TDC	Analog + ADC
Number of arrays	1	1	1
Number of devices	1,024 × 2,048	64 × 64	256 × 256
Inference voltage	Predet.	Predet. (array) 0.8V-1.0V(TDC)	Predet.
Need for calibration	Yes	Yes	Yes

Table 1.1: Comparison of the design choices of the Bayesian machine with leading emerging memory-based realizations of neural network hardware blocks. Abbreviations. RBM: restricted Boltzmann machine. MAC: multiply and accumulate. PCM: Phase Change Memory. ADC: analog-to-digital converter. CDS: correlated double sampling. SLC: single-level cell. TDC: time-to-digital converter. Predet.: Predetermined. ND: not discussed.

The situation is different for Bayesian inference. Unlike in neural networks, the multiply-and-accumulate operation is not needed, which limits the main benefit of analog computation and increases the energy cost due to periphery circuits. Moreover, higher precision is required for storing likelihood than for synaptic weights in neural networks. For instance, our machine uses eight-bit precision, which is not achievable with analog memristors. Therefore, our design relies on single-level bit cells read with extremely simple and highly energy-efficient precharge sense amplifiers (more details are provided in Chapter 2).

Our approach using sense amplifiers has several advantages over analog methods. The

sense amplifiers apply a current to the memristors only as long as necessary, while the analog approach needs to apply a current long enough for the voltages to stabilize, which usually involves a relatively slow feedback circuit. Furthermore, our approach avoids the need for energy-hungry analog-to-digital or time-to-digital converters, as the sense amplifiers naturally provide a digital output. This reliance on digital read with PCSA circuit distinguishes our work from other neural network implementations.

The digital read with PCSA circuit is highly flexible in terms of supply voltage, unlike analog approaches that require calibration and compensation mechanisms to eliminate circuit and device imperfections (e.g., voltage offsets due to circuit variability). Due to its differential nature, the sense amplifier functions over a wide range of voltage, without any need for recalibration or adjustment of any reference. This feature can be particularly useful in environments with little energy available (e.g., relying on variable energy harvesting) or in conjunction with dynamic voltage scaling. In contrast, neural network accelerators employing analog and/or mixed-signal circuitry usually require calibration and compensation mechanisms to eliminate circuit and device imperfections (e.g., voltage offsets due to circuit variability) [13, 186, 211, 213]. Overall, our simple read circuit does not need any calibration or compensation mechanisms, greatly simplifying circuit operation and increasing its flexibility in all types of conditions.

### 1.4.2 Other Bayesian Concepts Involving Nanodevices

	<b>This work</b>	Dalgaty et al. 2021[88]	Dalgaty et al. 2021[24]
Current status	<b>Fully HW</b>	Hybrid SW/HW exp.	Hybrid SW/HW exp.
Device	<b>HfOx memristor</b>	HfOx memristor	HfOx memristor
Use of the device	<b>Local digital memory</b>	Local analog memory	Local analog memory
Concept	<b>L-E Bayesian inference</b>	Bayesian learning	Bayesian NN inference
	Gao et al., 2021[215]	Vodencarevic et al., 2017[19]	Faria et al., 2018[216]
Current status	Simulated	Hybrid SW/HW exp.	Simulated
Device	Memristor	Stochastic MTJ	Stochastic MTJ
Use of the device	Local analog memory	RNG	RNG
Concept	Resilient NN inference	L-E Bayesian inference	L-E Bayesian inference

Table 1.2: Comparison of our work with approaches of the literature associating nanoelectronics with Bayesian concepts. Abbreviations. SW: software. HW: hardware. L-E: Low-Energy. NN: Neural Network. MTJ: magnetic tunnel junction. RNG: random number generation.

In recent years several works have explored connections between emerging memories (such as memristors) and Bayesian inference. Most of those works from the state of the art either re-



lied on computer simulations [215, 216], or on hybrid hardware/software realizations, where experimental nanodevices are used, and the rest of the system is simulated on a computer [19, 24, 88].

The works of [24, 88] focus on the implementation of Bayesian neural networks, a special class of Bayesian model that can model uncertainty much better than conventional neural networks, but do not feature the comprehensive explainability of the more traditional Bayesian inference addressed by our machine. These two works use memristors as main memory, as in our machine, and exploit the variability of memristors as a source of random variable. On the other hand, our machine focuses on reliability by eliminating the impact of memristor variability.

Bayesian inference in the work of [215] is used in a very different way. Unlike all other works reported in Table 1.2, the final goal of this work is to implement non-Bayesian machine learning model (a conventional neural network). By treating these networks as Bayesian neural networks modeling memristor variability, the authors are able to tolerate memristor imperfection better than more conventional approaches. In this work, memristors also implement memory.

References [19, 216] differ from all other works of Table 1.2, in that the nanodevices are not used as memory, but as random bit generators (replacing the linear feedback shift registers). These works are not full-system studies, as they do not address the memory question. Also, they focus on random variables with binary values, unlike our Bayesian machine that deals with multiple-valued inputs and outputs. On the other hand, we see at the end of the next chapter that incorporating some ideas from these works is a natural prospect for our Bayesian machine.

In our work on Bayesian Machines, we aim to advance the state-of-the-art of nanodevice-based Bayesian inference by developing fully fabricated Bayesian systems that improve the maturity of memristor-based Bayesian accelerators. These systems consist of locally distributed memory arrays that operate in parallel to perform Bayesian inference. Importantly, our approach differs conceptually from other proposals in the literature (Table 1.2), and here we aim to clearly define our contributions to the field:

- Our integrated circuits are the first fully fabricated memristor-based Bayesian inference systems.
- Compared with memristor-based neural network accelerators, our systems allow flexibility and simplicity (possibility to vary supply voltage, absence of calibration or compensation), and high robustness to read disturb (see Section 3.2.3), device variability, and outstanding robustness to single-upset events. All these features are due to the use of single-level cells read with particularly simple, robust, and energy-efficient precharge sense amplifiers.
- Our integrated circuits is a full system that features an array memory blocks that perform Bayesian inference in parallel, therefore solving the challenge of the distribution of the

---

various voltages to program and read memristors.

## 1.5 Conclusion

This chapter has presented the architecture of a Bayesian machine with distributed memristors that allows for local computation and minimal data movement for energy-efficient Bayesian inference. The unique requirements of Bayesian inference, which do not require multiply-and-accumulate operations like those commonly used in neural network accelerators, were taken into consideration in the design.

The goal of this work is to advance nanodevice-based Bayesian inference and develop fully fabricated systems that improve the maturity of memristor-based Bayesian accelerators. Collaboration between experts in Bayesian theory, memristor device modeling and characterization, and integrated circuit design is crucial for success. The project involves several steps, including theory development, design, fabrication, electrical characterization, and evaluation of real-world applications.

The Bayesian machine has the potential to be embedded at the edge with low power consumption, providing a practical solution for dealing with highly uncertain situations with little data and making predictions using an explainable mode. Additionally, the explainability of Bayesian inference is desirable in critical situations for ethical and regulatory reasons, and the Bayesian machine can recognize situations where it cannot provide a reliable answer, which could be useful for medical devices to prevent wrong decisions with serious consequences.

Overall, the developed integrated circuits, the stochastic Bayesian machine (see chapter 2) and the logarithmic Bayesian machine (see chapter 3) offer flexibility, simplicity, and high robustness to device variability and single-upset events. Our work on those projects has been published in a Nature Electronics article [14] and presented at the DATE 2023 conference [86].



## Chapter 2

# A Stochastic Bayesian Machine

Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin.

---

John VON NEUMANN

Real-world information, which consists of continuous-valued analog signals, requires conversion to digital data via quantization for digital circuits to process. However, high precision in quantization can be costly in terms of energy consumption and memory requirements. To address this issue, a recent trend in the literature has been to reduce precision and to implementing “approximate” computing.

These ideas have been massively applied to simplify coding schemes and optimize hardware for neural network-based algorithms in recent years. However, less attention has been devoted to the Bayesian approach, which involves successive product of probabilities. The main potential influences on energy consumption in Bayesian computation are access to probabilities data, multiplication operations between probabilities, and data movement. To address these challenges, we have employed stochastic computing, which uses random bit streams to perform computations and requires far fewer transistors and minimal data movement compared to traditional arithmetic.

In this chapter, we present a memristor-based stochastic Bayesian system that is fully implemented in hardware. Our prototype circuit incorporates 2,048 memristors and 30,080 transistors on the same chip, using a hybrid CMOS/memristor process. The architecture of the machine uses fully distributed memory, and due to the locality of computations and reliance on stochastic computing, minimum data movement is performed between different parts of the system. We provide a detailed explanation of the design, fabrication, and characterization of the Bayesian system, followed by an energy vs. accuracy performance study for a hand gesture recognition task. Our study shows that our system has an energy improvement of several orders of magnitude compared with a standard implementation of Bayesian inference on a microcontroller unit fabricated in a similar CMOS technology. Moreover, our system has an instant on/off feature due to the use of non-volatile memory and is inherently resilient to soft errors, making it suitable for use in extreme environments.

The primary energy expenditure in our machine was found to be due to random number generation. To mitigate this, we also considered using nanodevices for local generation of random bits, with the potential for farther energy reduction of our Bayesian machine. This led us to design and fabricate prototype circuits, which utilized unstable SMTJ devices and PCSA sensing circuitry for random bit generation.

The following Sections provide further details on the implementation and performance of our developed system.

This chapter, with the exception of section 2.5, is adapted from an article published in Nature Electronics [14].

## 2.1 Bayesian Inference with Stochastic Computing

An important challenge of the memristor-based Bayesian machine is that the classical computation circuitry, and particularly multiplications are normally an area-expensive operation in CMOS (Figs. 2.1a-b), raising a concern if a multiplier is associated with each likelihood memory array. For this reason, we rely on stochastic computing [207, 208]. Stochastic computing represents data as a bitstream of 1s and 0s, with the proportion of 1s representing the encoded data between 0 and 1. This unique method of representation limits all values in the stochastic approach to the range of 0 to 1. Integrating stochastic computing into the design of a memristor-based Bayesian machine with near-memory computing offers several advantages, particularly in terms of processing efficiency.

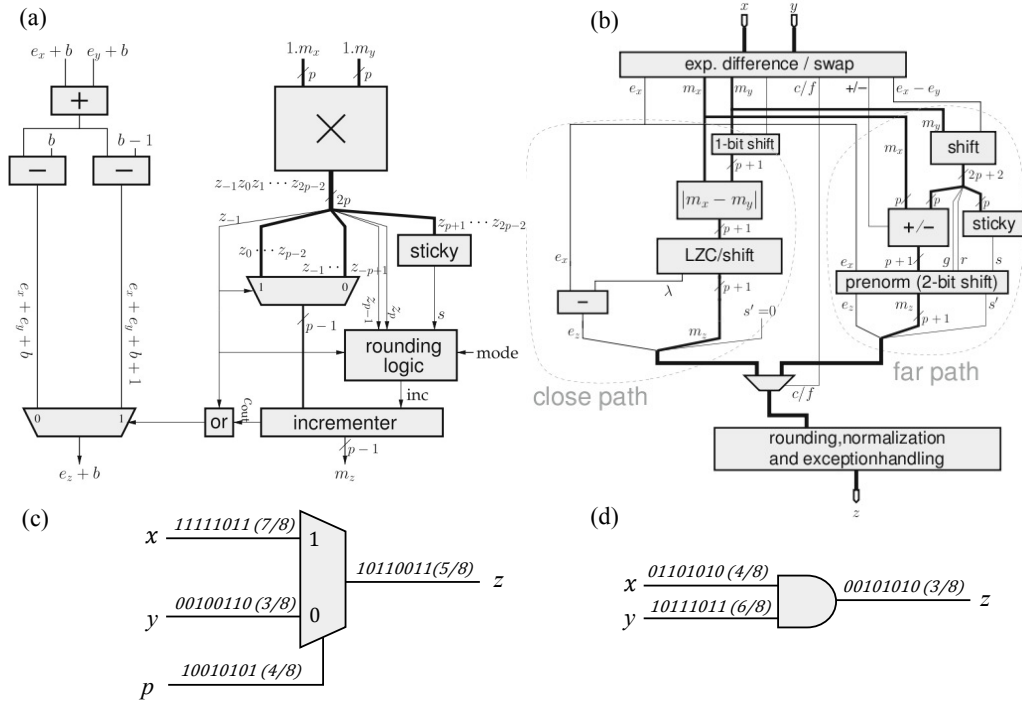


Figure 2.1: **Architectures of adders and multipliers.** Basic architecture of **a** a floating-point adder and **b** a floating-point multiplier. Both images are reproduced from [16]. **c** A simple multiplexer can perform the sum in stochastic computing, the output is  $z = px + (1 - p)y$ . If  $p = 1/2$ ,  $z = (x + y)/2$ . **d** A logical AND gate can perform the stochastic multiplication between two bit-streams. (Reproduced from [17])

The simple logic circuitry of stochastic computing operations, such as stochastic adders (as shown in Figure 2.1c) and stochastic multipliers (as shown in Figure 2.1d), is well-suited for co-location with memory circuitry. The multiplication of probabilities can be achieved using basic AND gates, resulting in minimal area cost [207] and efficient replication of the elementary processing circuit, the likelihood circuit in the Bayesian system. This enables the parallel

computation of a large number of computations, simplifying the overall design and increasing processing efficiency.

In addition, stochastic computing also reduces data size and data movement. While it may require a larger number of clock cycles compared to conventional calculation methods, the use of a single wire to encode data significantly reduces data size. An important aspect of this model of computation is that, as in most practical settings, probabilities tend to be low, and the output of stochastic computing AND gates is a zero value at most clock cycles. Therefore, the different blocks of the memristor-based Bayesian machine only need to pass single bits (Fig. 2.2) that are zeros at most clock cycles: the memristor-based Bayesian machine limits data movement considerably.

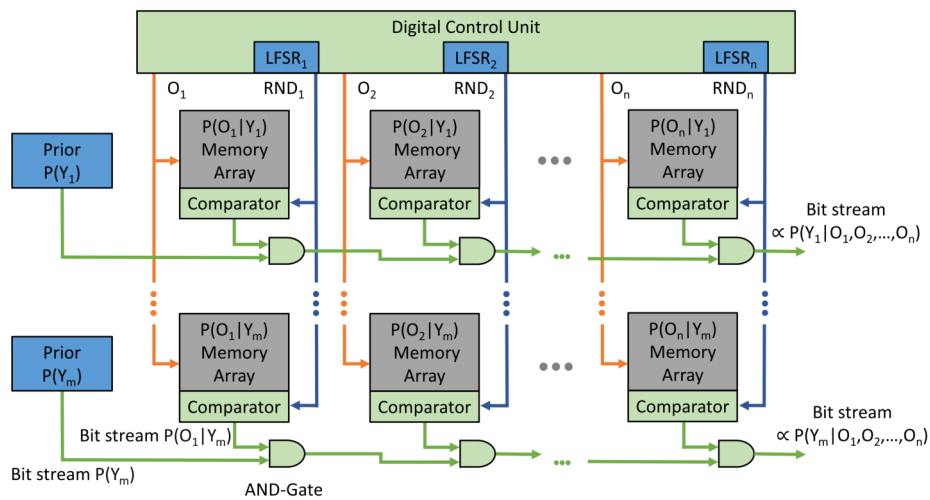


Figure 2.2: **General architecture of the Stochastic Bayesian machine.** Optimization of the Bayesian machine for hardware. Random numbers (RND) are generated using linear feedback shift registers (LFSRs), shared by column, and converted using digital “Gupta” circuits to a series of random bits proportional to the appropriate probability. Additionally, the likelihoods are normalized by the maximum likelihood value of the column to maximize the convergence speed of the machine. The stochastic multiplication is implemented by a single-bit AND gate.

The memristor-based Bayesian machine is an elegant concept, but its design faced significant challenges related to the use of stochastic computing. To ensure a consistent result on mathematical operations, stochastic computing requires high-quality random number generators that can encode uncorrelated bitstream data. Random number generators can be divided into two categories: PRNGs (Pseudo Random Number Generators) and TRNGs (True Random Number Generators). TRNGs derive random bits from physical sources that have intrinsic entropy, such as thermal noise, whereas PRNGs implement a deterministic system with a specific algorithm. For our Bayesian architecture, which uses near-memory computing, the generation of random numbers is a design choice that is subject to the algorithmic constraints of the

system.

For our stochastic Bayesian machine, we have chosen to employ digital pseudorandom number generators, specifically linear-feedback shift registers (LFSRs), to generate uniformly distributed numbers. In our design, a single LFSR is used per column (Fig. 2.2) so that each row can perform an independent stochastic computation, and different rows can rely on the same pseudorandom numbers. At each clock cycle, each likelihood block generates a random bit with the probability  $p$  by comparing the number generated by the vertical LFSR with the value of the probability  $p$  read from the likelihood memory array. After a defined number of clk cycles, the resulting bitstream has a ratio of 1 to 0 proportional to a probability stored in the memories. We use a special comparator circuit in our design, proposed by Gupta in 1988 [208, 217], for the comparator.

Additionally, stochastic computing converges slowly when multiplying low probabilities, and likelihoods in Bayesian models tend to be very low. To optimize the operation of the Bayesian machine without any loss of accuracy, we normalize the probabilities in a column so that the maximum likelihood in a column is one.



## 2.2 Design of the Stochastic Bayesian machine

### 2.2.1 The Big Picture

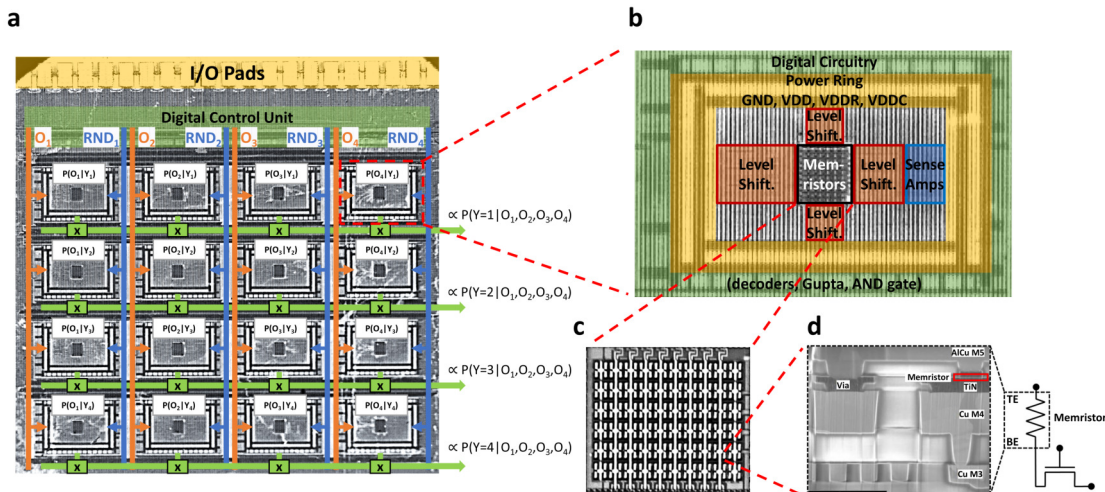


Figure 2.3: **Fabricated memristor-based Bayesian machine.** **a** Optical microscopy photograph of the Bayesian system die. **b** Detail of the likelihood block, which consists of digital circuitry and memory block with its periphery circuit. **c** Photograph of the 2T2R memristor array. **d** Scanning electron microscopy image of a memristor in the back end of line of our hybrid memristor/CMOS process. All subfigures use consistent color codes.

To validate the feasibility of memristor-based Bayesian inference, we designed and fabricated a prototype circuit in the hybrid CMOS/RRAM process described in Chapter 1. The CMOS part of the circuit is fabricated using a low-power foundry 130-nanometer process with four layers of metals. Hafnium oxide memristors are fabricated on top of the CMOS foundry layers, effectively taking the place of vias between metal layers four and five. Fig. 2.3a shows the fabricated die with the superimposed structure of the Bayesian machine. This test chip implements a system with 16 likelihood memory arrays, organized in four rows and four columns, connected through horizontal and vertical wires, and controlled by an on-chip digital control unit. As explained in Chapter 1, it was designed using an original semi-automatic homemade design flow.

The topology of the die follows closely the conceptual schematic of the Bayesian machine of Fig. 2.2. Figs. 2.3b-c show the details of one of the likelihood memory arrays and its periphery circuitry (Fig. 2.4a shows the associated schematic). Fig. 2.3d shows an electron microscopy image of a memristor integrated into the back end of line of the die. The hybrid CMOS/memristor fabrication process features some constraints due to its partially academic nature: only four levels of metals are available for interconnection (the fifth level being used only to access mem-

ristors). Still, this test chip allows us to demonstrate all the challenges associated with the fabrication of the Bayesian machine.

The first considerable challenge to design a reliable Bayesian machine is that memristors are prone to errors. Industrial applications of memristors use strong formal error-correcting codes (ECC) [218]. Using ECC in the Bayesian machine is inappropriate, as error detecting and correcting circuits would dominate both area and energy consumption if they needed to be replicated for each likelihood memory array [219]. Therefore, we use an alternative strategy: memristors are used as single-level cells, and bits are programmed in a complementary fashion, and read differentially by sense amplifiers comparing the resistance of two memristors (see Fig. 2.4b-c and their description below). This technique has been shown previously to reduce errors as efficiently as single error-correcting, double error-detecting codes (extended Hamming), using the same degree of memristor redundancy, and necessitating no error decoding circuit [26]. It is used here within a full system for the first time. The likelihoods themselves are coded in binary representation as eight-bit integers.

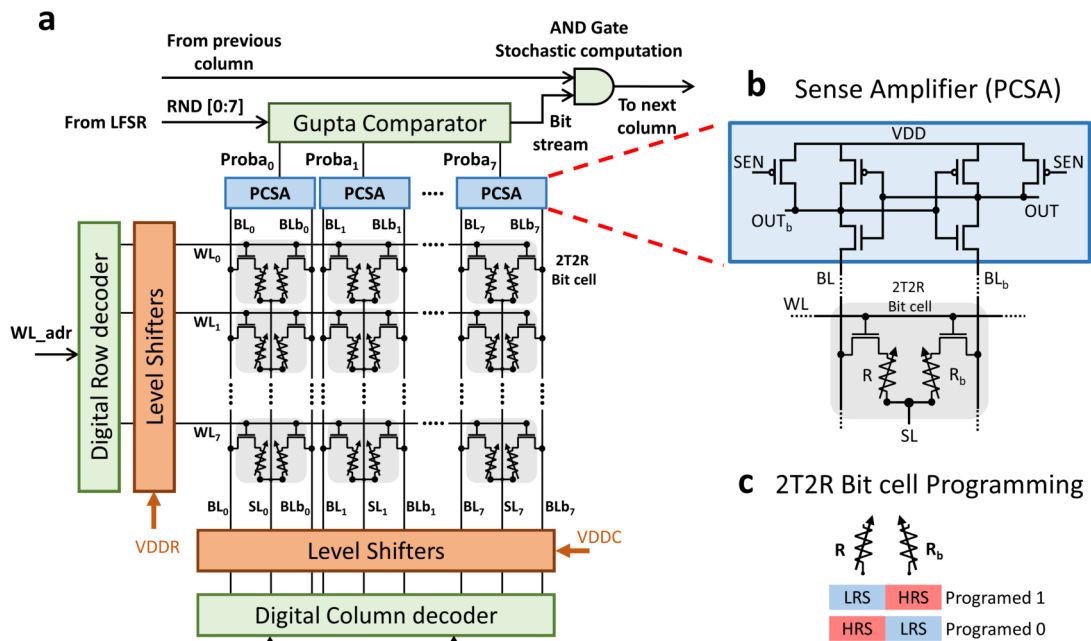


Figure 2.4: **The designed memristor-based likelihood circuit.** **a** Schematic of the likelihood block presented in Fig. 2.3b. **b** Schematic of the differential precharge sense amplifier used to read the binary memristor states. **c** Principle of complementary programming of the 2T2R bit cell memristors.

Programming the memristors within the Bayesian machine is a second major challenge. The nominal voltage of our foundry CMOS process is only 1.2 volts for the digital functions, whereas the forming and programming operations of the memristors require several volts. The distribution of the higher-than-nominal voltages is a challenge in systems such as this one with

massively distributed memory blocks that each need access to all voltage supplies. For this reason, in our circuit, the signals controlling the forming and programming operations are distributed as nominal-voltage logic signals. They are raised to higher voltages by level shifters distributed locally all around the memristor arrays. Fig. 2.5c-d shows the schematic of the level shifters implemented in our circuit. **Section 2.2.2 details specifically how the memristors can be formed and programmed to zero or one state using these level shifters.**

In each likelihood memory array, each row includes the eight bits of one likelihood value, and the different rows correspond to the different values of the observation. When new observations are presented to the system, a whole row needs to be read in each of the 16 likelihood memory arrays (the observations presented through vertical wires, acting as a row address for each memory array see Fig. 2.2). Fig. 2.4b shows the differential precharge amplifier used for reading the memory bits. This circuit precharges the two complementary bitlines to the supply voltage, and naturally detects the bitline that discharges the fastest using cross-coupled inverters. This very compact circuit is highly energy-efficient, as it involves no direct current between ground and the supply voltage. It is also highly robust due to its differential nature and requires no calibration. Each column of each likelihood memory arrays features a precharge sense amplifier, and during a read operation all 128 sense amplifiers of the Bayesian machine function simultaneously (corresponding to the simultaneous read of 8 bits for each of the 16 likelihood memory arrays). **Section 2.2.3 explains in more details the operation of precharge sense amplifiers.**

After this read operation, the Bayesian machine can perform Bayesian inference using stochastic computing. The four LFSRs generating pseudorandom numbers each clock cycle are situated within the digital control unit and their output are distributed to the likelihood memory arrays through vertical wires. Each likelihood block uses the GUPTA comparator circuit to generate random bits with a probability equal to the value read in their likelihood memory array, based on the output of the LFSRs. This circuit gives equivalent results than a comparator, but with lower area cost [217]. The resulting bits are combined by AND gates, all happening during a single clock cycle. An important aspect of the design is the clock is only necessary in the digital control circuitry block, outside the core machine, to operate the LFSRs. This feature limits the energy cost of clock distribution. **Section 2.2.4 presents in more detail the operation of the Bayesian machine.**

## 2.2.2 Programming methodology of the Bayesian machine

Memristors are programmed with voltages higher than the nominal voltage used for digital circuitry. The higher-than-nominal programming voltage requirement of memristors is a minor concern in systems that separate memory from computing, as a single dedicated high-voltage circuitry can be associated with the memory array. The Bayesian machine, by contrast, features multiple small memory arrays fully embedded within logic. The programming of memristors, therefore, requires the distribution of higher-than-nominal programming voltages locally and

an appropriate programming strategy. Managing this complexity is the largest design challenge for obtaining a functional Bayesian machine.

The Bayesian machine stores bits using a 2T2R structure (see Fig. 2.6a): the bit cell is composed of a “bit line” and a “bit line bar” memristor ( $R$  and  $R_b$ ), positioned on the same row, and each associated with a selection nMOSFET. The two memristors are connected to two different bit lines (BL and BLb) on their bottom electrode side. Conversely, the top electrode of the two memristors is connected to the same source line (SL), to limit wiring and programming circuitry. This shared source line requires careful attention when programming the memristors. The gates of the control transistor of all memristors on the same row are connected to the same word line (WL).

Programming operation is controlled by nominal-voltage signals (CBL, CSL, and CWL) and an address, all provided by the digital control unit of the Bayesian machine. Decoders select the addressed row and column (Fig. 2.6a). Then, local level shifters apply either ground or higher-than-nominal voltages (VDDR and VDDC) to the crossbar arrays, where needed:

- Each memory row features one level shifter (LS, commanded by CWL), which controls the word line that feeds the gates of nMOS selection transistors of the memory rows. Depending on the value of CWL, the level shifter of the addressed row connects its word line either to ground or to the VDDR power supply (see Figs. 2.5b and c).
- Correspondingly each memory column features two level shifters (regular LS, commanded by CSL and tri-state TLS, commanded by CBL), which control the source lines and the bit lines. These level shifters connect the source line and the two bit lines either to the ground or to the VDDC power supply. Notice that the two-bit lines BL and BLb are connected to the same level shifter, and therefore always receive complementary voltage (see Figs. 2.5b and d).

When the system is first characterized, the memristors need to be formed. Fig. 2.6a shows the voltages that need to be applied on the memristors during the forming step. The Table in Fig. 2.6c lists the associated CSL, and CBL values. Note that when one memristor is being formed, the other memristor is unaffected, due to the fact that the bit line and bit line bar always see complementary voltages (as they are connected to the complementary outputs of the BL level shifter). Once the proper address, CSL, and CBL values have been set, CWL is raised to one, and forming occurs during the CWL pulse (of one-microsecond duration, see Fig. 2.6b).

Once the memristors are formed, they are all in a low resistance state. The memristors can then be programmed and reprogrammed at will, either in low resistance state (LRS) by a SET operation, or to a high resistance state (HRS) by a RESET operation. In our 2T2R bit cells, the two memristors are always programmed in a complementary fashion: this complementary technique allows high robustness to memristor variability and read disturb effects [26, 220].

To program a zero value in the bit cell, the bit line memristor is programmed to high resistance (RESET) and the bit line bar memristor to low resistance (SET). The opposite is done to

program a one value. Fig. 2.6a shows the required voltages to program one and zero values, and the Table in Fig. 2.6c lists the corresponding CSL and CBL value. The voltage levels of VDDR and VDDC used for the SET and RESET operations are shown in Fig. 2.6b.

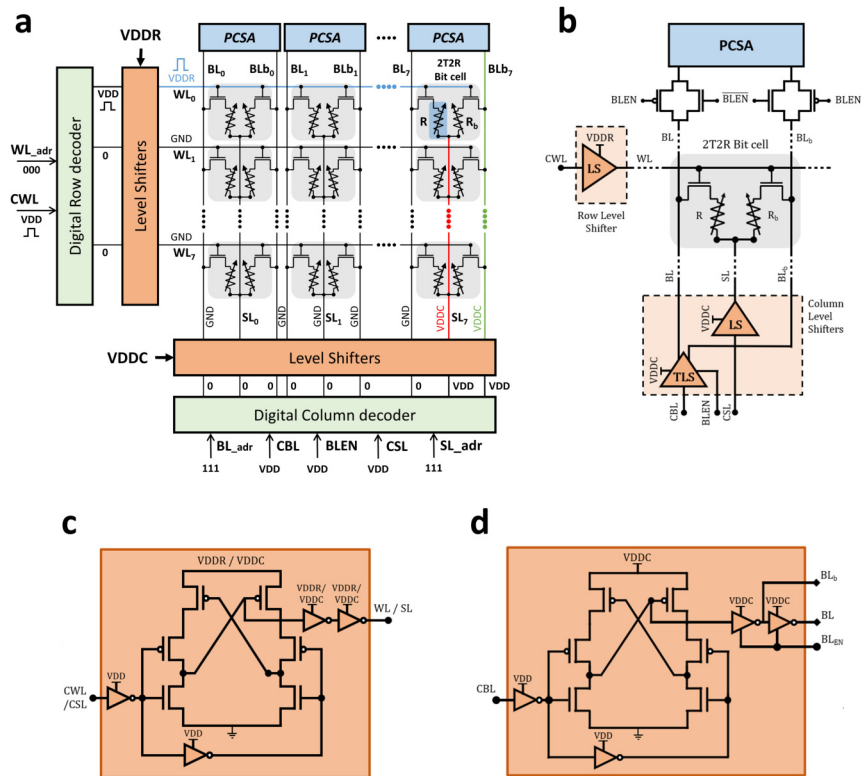


Figure 2.5: **Programming circuitry for the likelihood memory arrays.** **a** Detailed schematics of the likelihood memory array, with its programming and reading periphery circuitry, displaying the voltages needed to perform a SET operation on the first row, last column left memristor  $R$ . **b** Schematics of the 2T2R bit cell connections to the reading and programming circuitry. Two level shifters (conventional level shifter LS and three-state level shifter TLS) and one sense amplifier (PCSA) are implemented in each column. One level shifter is implemented in each row. The digital signal BLEN allows choosing between the reading or programming mode. **c** Transistor-level schematics of the level shifter (LS) and **d** the three-state level shifter (TLS) circuits.

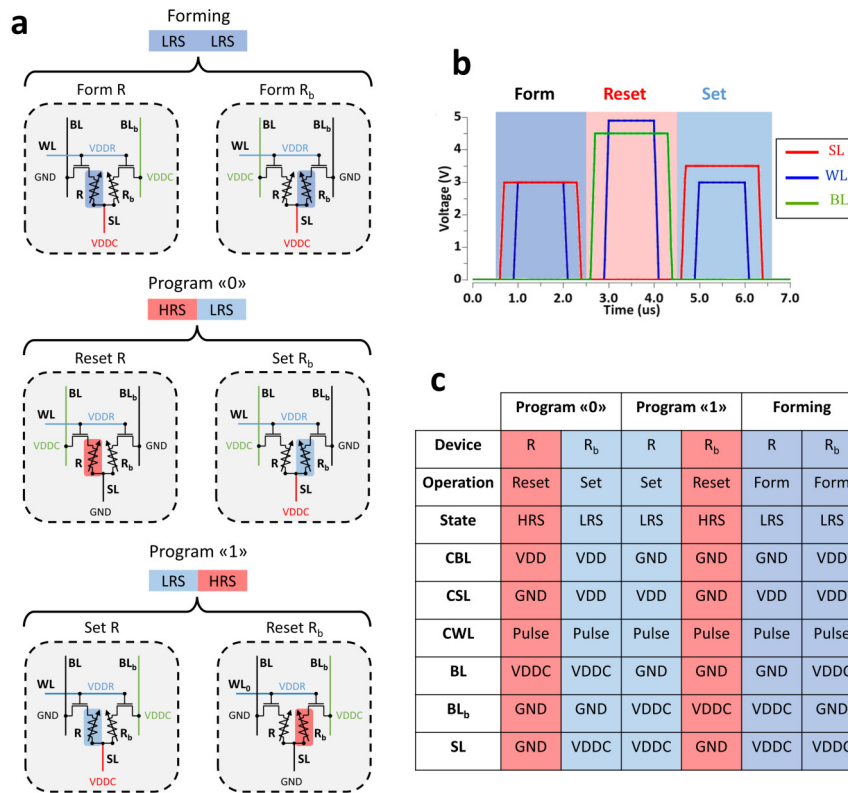


Figure 2.6: **Programming methodology for the 2T2R bit cell.** **a** Voltages that need to be applied on the bit line BL, bit line bar  $BL_b$ , and source line SL for forming, programming a zero, and programming a one in a 2T2R Bit cell. **b** Programming voltage levels and timings used for the Forming, RESET, and SET operations (mentioned in the Methods section of the main article). **c** Table summarizing the configuration of programming signals (level shifters) for the different programming operations supported by the memory array (forming, programming a zero, and programming a one).





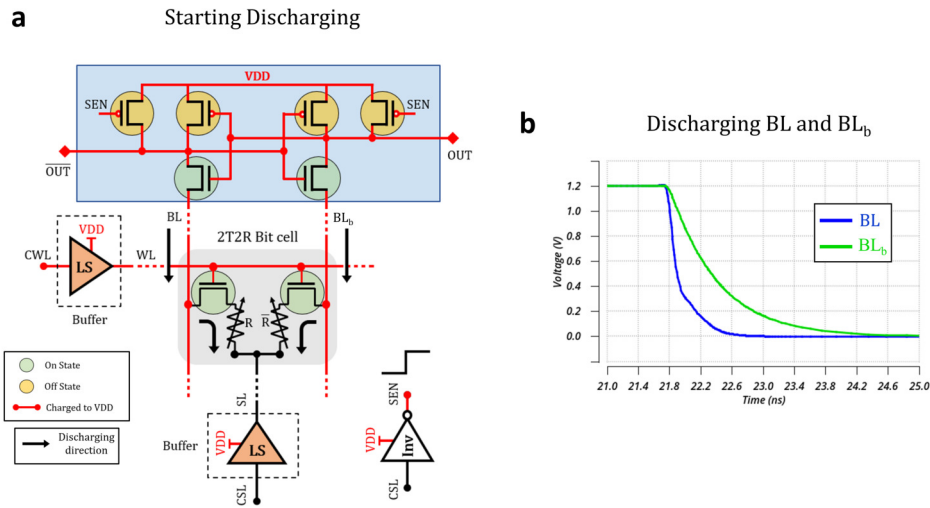


Figure 2.8: **Read operation for the likelihood memory array: discharge phase.** **a** Schematic and **b** circuit simulation of the discharge phase, BL and BL<sub>b</sub> are discharged to GND with different speed.

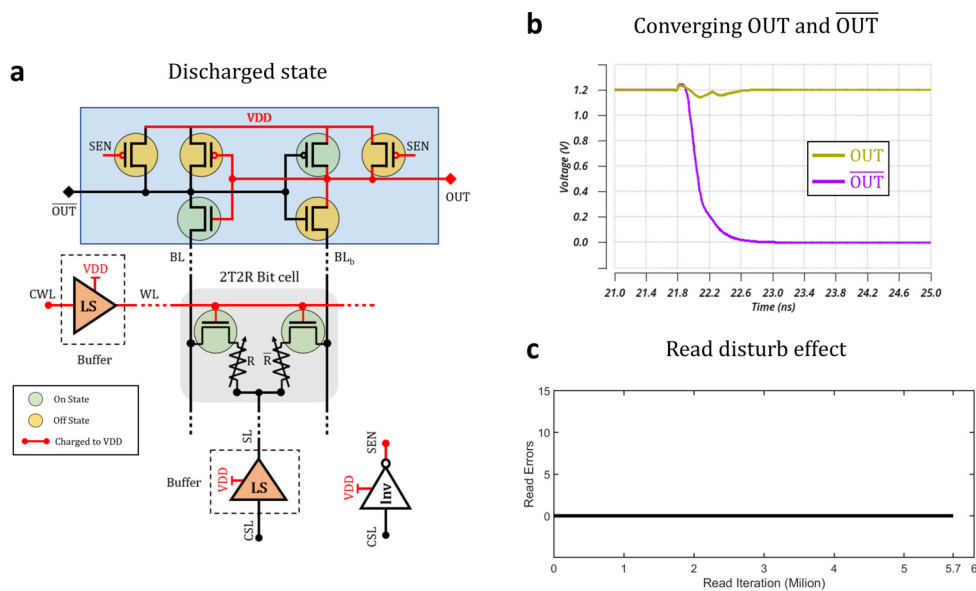


Figure 2.9: **Read operation for the likelihood memory array: Precharge phase.** **a** Schematics and **b** circuit simulation of the outputs of the PCSA converging to a stable state, while BL and BL<sub>b</sub> are completely discharged to GND. **c** Experimental measurement of the read disturb on likelihood memory arrays, with a VDD value of 1.2 volts. Even after 5.7M read operations of the whole array, no error is seen.

### 2.2.4 Inference Using a Bayesian machine

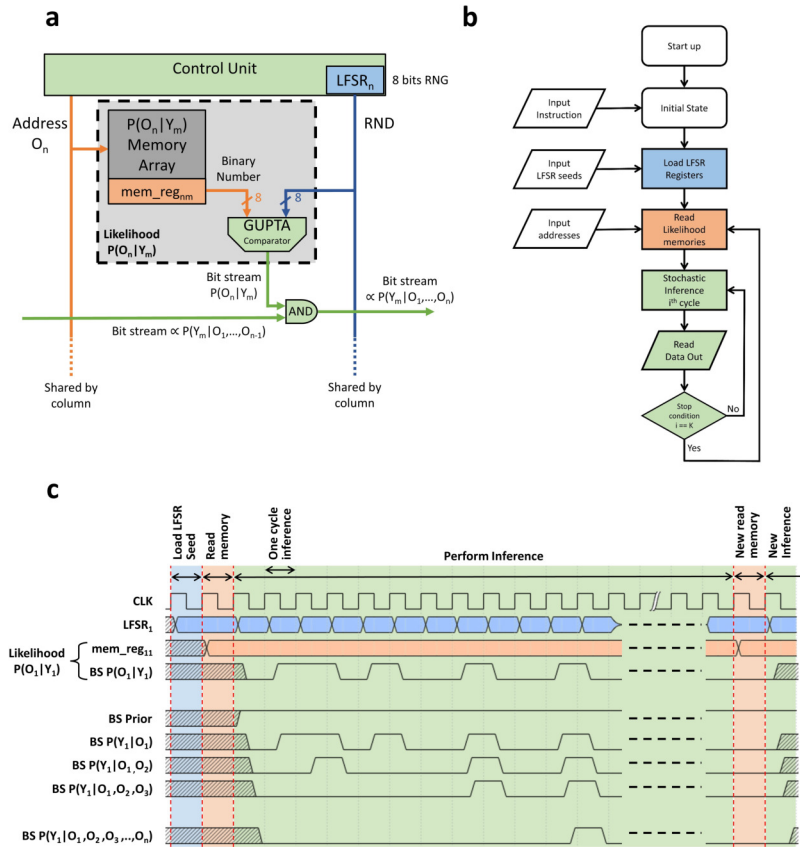


Figure 2.10: **Detailed operation of the Bayesian machine.** **a** Schematic illustrating the detailed architecture of a likelihood elementary block. **b** Flowchart of the different operations to perform a Bayesian inference computation in the Bayesian machine. **c** Time diagram illustrating the operation of the Bayesian machine.

To understand the inference operation of the Bayesian machine, we focus more on the design and functionality of the likelihood elementary block and its detailed schematics (taking an example of likelihood at row  $m$  and column  $n$ ). The function of this block is to output a bit stream with a probability proportional to the likelihood  $P(O_n|Y_m)$ . Fig. 2.10a shows the detailed schematics of this elementary block, which has two input vectors, one input representing the observations  $PO_n$  in form of memory addresses of probability values, and another input of 8 bits random numbers generated by LFSR. The likelihood block consists of: an  $8 \times 8$  memristor array of  $2T2R$  structure, including all read and program circuitry, and a GUPTA comparator circuitry. In addition, Fig. 2.10b shows a flow chart of the stochastic Bayesian inference in the Bayesian machine, and Fig. 2.10c shows a time diagram of the machine operation. The color code throughout Fig. 2.10 is as follows: the blue color corresponds to random number genera-

tion, the orange color to memory read, and the green color to the actual stochastic inference.

Before starting Bayesian inference operations, we first need to load the LFSR seeds. In our design, the input seeds are loaded from input pads and routed to the four LFSRs of the Bayesian machine by the Bayesian machine digital control unit. Being able to choose the LFSR seed is important for our study (see section 2.3.3). In a final design, optimal LFSR seed values could be loaded automatically by the control unit. As the LFSRs have a periodical output, the seed initialization needs to be performed only once as long as the digital power supply VDD remains on.

The Bayesian machine then performs inference in two main phases:

- **Memory read.** Likelihood values are read from the memristor arrays, based on the input observations (acting as row addresses). Observations  $O_1, \dots, O_n$  are off-chip inputs loaded from dedicated input pads, then addressed to the likelihood memory arrays by the Bayesian machine digital control unit (one observation for each column, see Fig. 2.2). All likelihood memory arrays are read simultaneously.
- **Iterative stochastic inference phase.** At each clock cycle, LFSRs generate new eight-bits pseudo-random numbers. These numbers feed the Gupta comparator circuits to compute the stochastic bits based on the read likelihood values. Outputs of Gupta comparator circuits from the same row are fed to a chain of AND gates, to perform the stochastic multiplications; the results of those multiplications represent the Bayesian machine outputs. All these operations are performed by purely combinational circuits in one clock cycle. As the periodicity of the LFSRs is 255 cycles, the maximum number of iterations is 255 cycles.

The Bayesian machine can operate in two modes. In the conventional stochastic inference mode, computation is performed for a pre-chosen number of cycles. The machine decision is based on bit streams counting of outputs (the number of generated one values). In the “power-conscious mode”, the Bayesian machine stops the stochastic inference iterations as soon as one of the outputs produces a bit value of one; the decision is made based on that result (see section 2.4).

## 2.3 Characterization of the Stochastic Bayesian Chip

Stochastic Bayesian machines hold great promise as a means of implementing Bayesian inference tasks, and as such, it is imperative to experimentally verify the reliability of memristor-based chips for such computations. In this section, we present a series of on-chip experiments and simulations that aimed to optimize the performance of a stochastic Bayesian machine. To this end, we conducted most of our experiments using non-packaged dies, employing a custom-made 25-pads probe card for probe testing (shown in Chapter 1), with the exception of read-disturb experiments which were conducted on packaged dies.

We first emphasize the importance of exploring the forming and programming conditions of memristors, given their process-to-process variability and generation-to-generation changes. The fabrication of new memristor-based chips with improved processes renders the forming and programming voltages potentially different from those of previously tested chips. **These first results are reported in section 2.3.1.**

Subsequently, we conducted an experimental study on the effect of LFSR seed choices on the on-chip inference results following successful programming of the likelihoods in the memristor arrays. **These measurements are shown in section 2.3.2.** Our findings underscore the considerable improvement achieved with optimal seeds as compared to the imperfections and deviations observed with non-optimal seeds.

To optimize the performance of the stochastic Bayesian machine, we carried out stochastic Bayesian inference simulations aimed at searching for optimal seeds for the linear feedback shift registers. **This exploration is detailed in section 2.3.3.**

### 2.3.1 Forming, Programming, and Read-Disturb Experiments

The first results of the electrical characterization of the test chip are presented in Fig. 2.11. Fig. 2.11a shows the measured results of reading the likelihood memory arrays of the chip, before forming and programming, naturally showing random values (as bits are stored in a complementary manner using two memristors, reading the memory with unformed bitcells leads to random results). Fig. 2.11b presents the same measurements after programming: the intended patterns are obtained without errors, showing the efficiency of the complementary programming technique. These artificial patterns were chosen so that performing Bayesian with random inputs allows exploring the whole range of possible output probabilities, allowing to test the functionality of the demonstrator.

The programmed bits are very stable. When remeasuring the die five months after programming, no error was seen. The demonstrator was stored at room temperature during these five months

The typical operation of a Bayesian machine requires frequent read operation on the likelihood arrays, with rare reprogramming: the memristors need to be reprogrammed only when

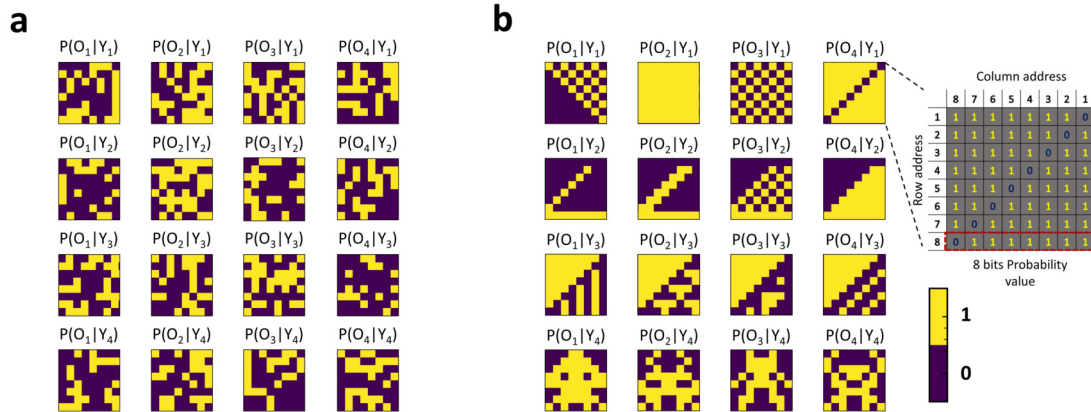


Figure 2.11: **Measurements of the fabricated memristor-based Bayesian machine.** **a** Measurements of the likelihood stored in the memristors, before they have been formed. As the bits are programmed in a complementary fashion involving two memristors, the result of the measurement appears random. **b** Measurements of the likelihood stored in the memristors, after they have been formed and programmed. No bit error is seen.

the model is changed (e.g., after a recalibration based on new training data). It is therefore critical that read operations cannot accidentally change the state of the memory devices (read disturb effect).

To evaluate the existence of read disturb, we performed repeated read operations on one likelihood memory block of the fabricated memristor machine. The read operations are performed by the on-chip precharge sense amplifiers. We used a digital power supply voltage of 1.2 volts, as it is the highest digital supply voltage supported by our system, and the more at-risk of read disturb effects. Our experimental setup allows reading the likelihood array approximately one million times per day. We observed that after five days of continuous read operations, i.e. a total of 5.7 million read operations of a complete likelihood memory array, no likelihood bit had changed (see Fig. 2.9c), suggesting a high immunity to read disturb effects.

The immunity to read disturb of our machine can be explained by three main reasons:

- Hafnium oxide memristors are naturally resilient to read disturb effects due to the highly nonlinear nature of their switching process [223].
- The complementary 2T2R approach not only reduces the impact of device variability, but also read disturb effects. Even if the read disturb effect increased the resistance of a low resistance state device, the stored likelihood would be affected only if the disturbed device ends up in a resistance higher than its complementary high resistance device.
- The precharge sense amplifier used to read the devices naturally mitigates read disturb effects. When the sense amplifier has identified the stored bit, the nodes connecting the

sense amplifier to the memristor array are rapidly pulled down to the ground, and the read memristors therefore see zero voltage. Therefore, current is applied to the memristors only during the time needed for the sense amplifier to differentiate between the two possible memory states. This mode of operation contrasts with conventional current-mode sense amplifiers where current is applied during a fixed time that has to be chosen in a worst-case scenario [224].

### 2.3.2 Bayesian Inference Experiments

The actual results of the stochastic Bayesian machine operation are shown in Fig. 2.12a-b. In Fig. 2.12a, the LFSRs were initialized using random seeds. The x-axis represents the theoretical result expected from Bayes' law (i.e., the desired output for the Bayesian machine), while the y-axis represents results measured experimentally by counting bits at the output of the die. The different points in the Figure are obtained by randomly changing the inputs  $O_1$ ,  $O_2$ ,  $O_3$ , and  $O_4$  of the circuit, and by changing the nominal supply voltage  $V_{DD}$ . For each random set of input, the system was operated 255 clock cycles (i.e., the periodicity of the LFSRs). The number of ones outputted at each row is counted and divided by 255 to be converted as a probability, and plotted in the Figure.

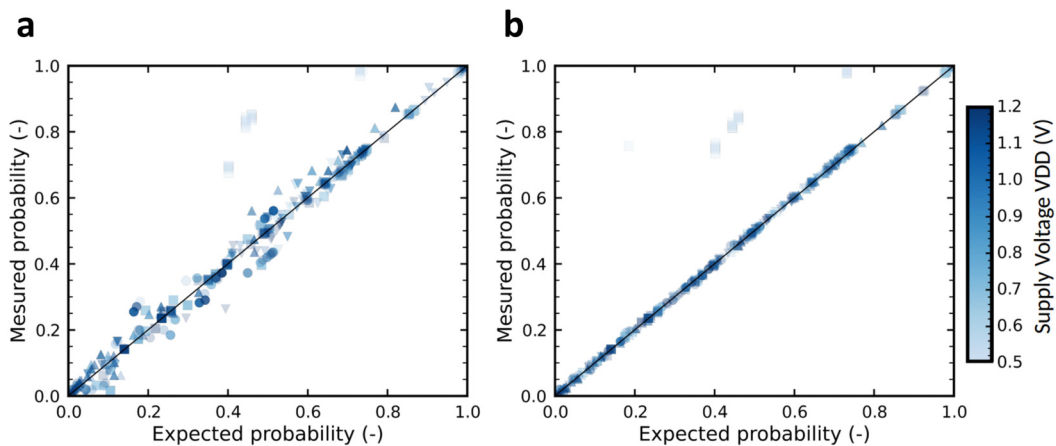


Figure 2.12: **Measured output of the Bayesian machine.** **a** measured posterior probability as a function of the expected value from Bayes' law. The different points correspond to random observation inputs. The different rows are pooled in the same graph. The points are obtained with various supply voltages  $V_{DD}$  ranging between 0.5 and 1.2 volts. This graph is obtained with non-optimal LFSR seeds. The measured probabilities are obtained by averaging the experimental measurements over the full LFSR period (255 cycles). **b** Same as **a**, using optimal LFSR seeds. The symbols indicate which row of the Bayesian machine was used (circle, up triangle, down triangle, square: first, second, third, and fourth row).

In Fig. 2.12a, we see that the measured probabilities closely follow Bayes' law, with some deviation. This deviation can be attributed to the imperfect nature of LFSR-generated pseudo-

random numbers. The numbers generated on the different columns have correlations, which prevent stochastic computing from being perfectly accurate.

Fortunately, this imperfection can be avoided by an intelligent choice of the LFSR seeds. In the measurements of Fig. 2.11b, the chip was initially programmed by an optimal seed choice. We see that the measurements follow Bayes' law perfectly for all possible inputs, which highlights the high potential of stochastic computing for Bayesian inference. Section 2.3.3 of this document details how the optimal LFSR seeds were chosen, their value, and the reason for their existence. This result shows that the Bayesian machine is able to produce accurate outputs, despite its reliance on very simple pseudorandom numbers. The methodology for finding this optimal seed values is presented in the next subsection.

In addition to its limited data movement, which we analyze further in the section 2.4, our approach offers two significant opportunities for low-energy operation. First, as the likelihoods are stored in non-volatile memristors, the system provides an instant on/instant off feature. Therefore, the power supply can be turned off any time the system is not used. Second, due to its fully digital nature, the system is flexible in terms of supply voltage. Figs. 2.11a-b highlight that the system remains fully functional when reducing the power supply down to a value of 0.6 volts, although the nominal supply of our CMOS technology is 1.2 volts. This operation allows reducing power consumption by a factor of approximately four. At lower voltages (light blue points), the Bayesian inference becomes less accurate. This voltage limit is due to the threshold voltage value of the thick oxide transistors used within the memory array, around 0.6 volts. Even lower supply voltages could therefore be used by using lower threshold-voltage transistors.

### 2.3.3 The search for Optimal LFSR seeds

In its original form, stochastic computing relies on high-quality and non-correlated random numbers. Generating high-quality random bits and maintaining the independence of stochastic bits after they have been processed by stochastic computing circuits is a major challenge, leading to costly strategies like randomness isolation and regeneration [225]. However, in practice, multiple works have shown that pseudorandomness and correlations are not necessarily a fundamental issue for stochastic computing, if they are properly considered in the system design [208]. In the particular case of our Bayesian machine, we found that we could rely on low-quality, low-cost LFSR-generated random numbers, and still get accurate Bayesian inference, under the condition of initializing the LFSRs with well-chosen seeds.

Our design exploits one eight-bit LFSR per column, which generates pseudorandom numbers with a periodicity of only 255 clock cycles. Fig. 2.13a highlights the correlation between the random numbers generated by the LFSRs, by plotting the 255 values generated by each LFSR as a function of the 255 values generated by each other LFSR at the same time. This Figure is plotted with initially random-chosen LFSR seeds, as in Figs. 2.11a. Obvious correlations are observed between some LFSR. These correlations cause some inputs to the stochastic computing

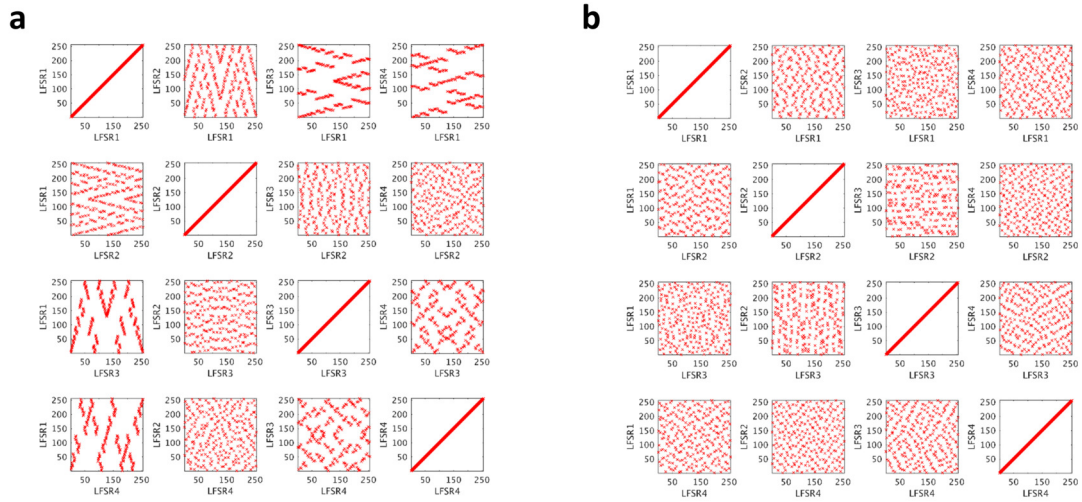


Figure 2.13: **Correlations of the random numbers generated by four LFSRs.** Each graph presents the output of one of the four LFSRs of the Bayesian machine, as a function of the output of another LFSR. Each graph contains 255 points corresponding to the 255 cycles of operation of the Bayesian machine. Graphs on the diagonal (LFSR1/LFSR1, LFSR2/LFSR2, LFSR3/LFSR3, LFSR4/LFSR4) appear as  $x=y$  lines, by definition. **a** The random numbers generated with initially randomly chosen seeds used in in Fig. 2.12a. The presence of very discernible patterns in some of the graphs (LFSR1/LFSR3, LFSR1/LFSR4), indicates the existence of a strong correlation between the output of some LFSRs. On the other hand, the outputs of some LFSRs appear largely uncorrelated (LFSR1/LFSR2, LFSR2/LFSR3, LFSR2/LFSR4). The seeds for the four LFSRs are, respectively, in hexadecimal representation: 50, E9, 10, and C6. **b** The random numbers generated with the optimal seeds used in Fig 2.12b. The results show an absence of evident correlation between all outputs of the different LFSRs. The seeds for the four LFSRs are, respectively, in hexadecimal representation: EB, FB, 7E, and 5C.

AND gates of the Bayesian machine to be correlated, leading to the deviations between Bayes' law and measurements in Figs. 2.11 a.

This type of correlation is observed with most choices of LFSR seeds. However, the choice of seeds used to generate Fig. 2.13b, referred to as “optimal” throughout the thesis, shows dramatically reduced correlations. This allows obtaining the highly accurate results of Figs. 2.11b, and suggests that these seeds should be used for all computations. Due to the periodicity of the LFSR, LFSR initialization needs to be performed only once, when the system is turned on. The gesture recognition task, which we analyze further in the next section, requires six LFSRs. We also used eight-bit LFSRs and optimized the seed choice for this situation.



## 2.4 Energy efficiency of the Stochastic Bayesian machine

Our test chip allows validating the possibility to address the challenges of designing and fabricating the memristor-based Bayesian machine. This system is, however, not adapted to evaluate the power consumption of a final system, as the test chip is too small to implement real-life applications. Additionally, the constraints of the semi-academic process and the wide transistors that we employed cause a high increase of the dynamic capacitive energy consumption. To evaluate power consumption, we switch to a larger design and a realistic application, and use industry-standard integrated circuit design tools to assess energy consumption with a fine granularity.

We focus on an application of gesture recognition. The input to the Bayesian machine is a selection of features extracted from the time traces on an inertial measurement unit (IMU, see Appendix section of this chapter). The goal of the system is to recognize the hand gesture performed by a user wearing the IMU (see Fig. 2.14a): the gesture of writing the digit one, the digit two, the digit three, or a signature (see Appendix section of this chapter). This task is performed by a scaled-up version of the Bayesian machine, using 24 (six columns, four rows) four-kilobits likelihood memory arrays.

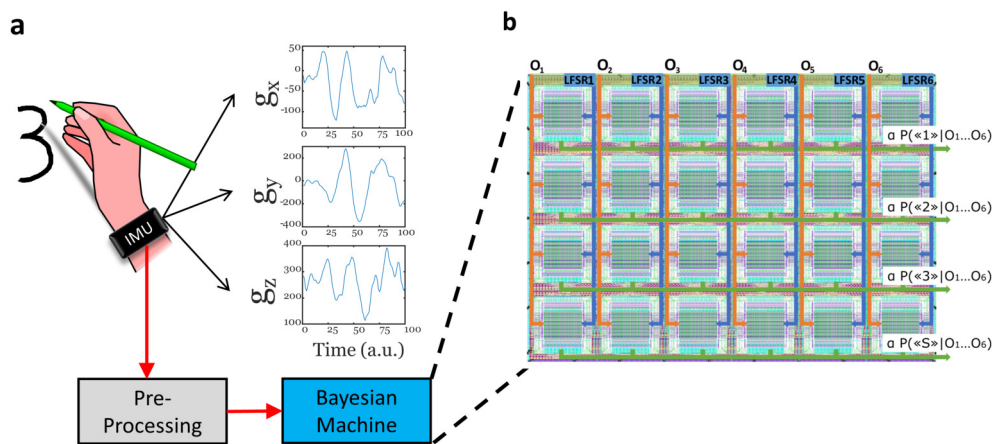


Figure 2.14: **Application of the Bayesian machine on a practical gesture recognition task.** **a** Setup with inertial measurement unit used to record the gesture recognition dataset. **b** Masks of the placed-and-routed Bayesian machine design used to perform the design-level gesture recognition analysis.

We designed and laid out this system in our reference process (low-power foundry 130-nanometer process) and evaluated its energy usage based on simulations. We can see in the image of the masks shown in Fig. 2.14b that in this scaled design, the area of the memristor arrays is now dominant, with regard to the memory periphery circuitry and the wiring of the Bayesian machine. The energy consumption is based on an exact scenario using value change dump files, and required adapting the standard flow of energy analysis, which is not naturally

adapted for systems where the memory is as distributed as ours (see Appendix section of this chapter). The energy consumption of the memory arrays and the digital circuits is evaluated independently using circuit (Spice) simulation and digital circuits analysis tools. In both cases, parasitic capacitances were extracted based on a complete layout and were included in the energy analysis (see Appendix section of this chapter).

Fig. 2.15a shows the energy consumption of the different elements of the system in the three operation phases (after the likelihoods have been programmed). The LFSR initialization consists in loading the seeds of the six LFSRs of the circuit. This operation consumes 0.38 nanojoules; it needs to be performed once when the system is turned on, and does not need to be repeated as long as the power remains on. It, therefore, remains a minor contribution to energy consumption. This energy could also substantially be reduced by hardwiring the value of the optimal seeds (whereas seeds are loaded from external inputs in our design). By contrast, the memory read operation needs to be performed each time a new input is presented to the system, and consumes a total of 0.3 nanojoules, including both the energy associated with the memory circuit themselves and the digital control circuitry. The actual stochastic inference, corresponding to the stochastic computation, consumes 2.2 nanojoules (assuming that all 255 cycles of the LFSRs have been operated). Therefore, in sharp contrast with von Neumann-type architectures [162], the energy consumption of the computation is dominant with regards to the energy for accessing data, highlighting the benefits of computing close to memory.

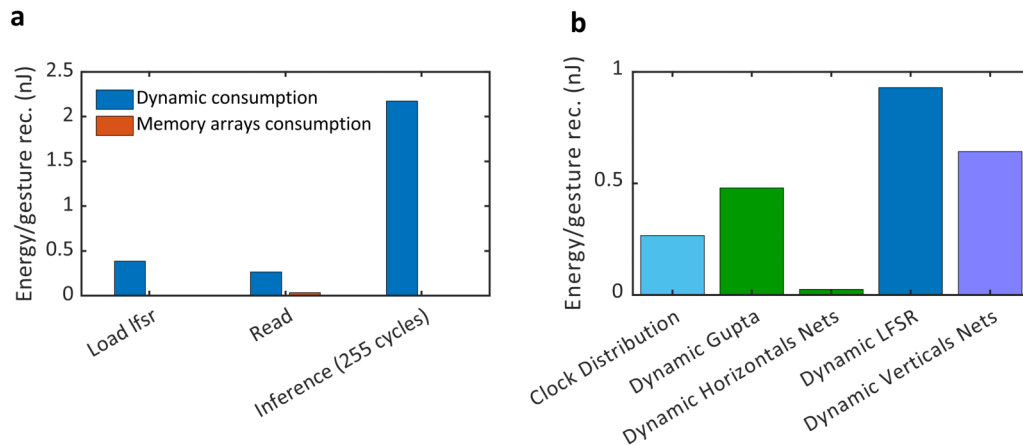


Figure 2.15: **Energy analysis of the Bayesian machine on a gesture recognition task.** **a** Energy consumption of the system (Dynamic consumption and memory arrays) during the three phases of computing: loading the seeds into the LFSR, reading the memories, and the actual inference of 255 cycles. **b** Energy consumption of the system's important points during the inference phase for 255 cycles. All energy numbers are given for a supply voltage of 1.2 volts.

To further analyze the energy usage of our architecture, Fig. 2.15b details the different sources of energy consumption during the stochastic computation. The clock distribution represents 11% of the energy consumption. This number remains relatively modest, because the clock is

not distributed within the Bayesian machine itself, but only to the external digital control circuitry. The probability multiplication itself (AND gates) and the transfer of the output of the blocks to the next block, through horizontal wires, represent only 1% of the total energy consumption. Random number generation (LFSRs, Gupta circuits) represents 60% of the energy consumption, and their distribution through vertical wires 28%. Future efforts should therefore focus on this part (see Section 3.5 and Conclusions).

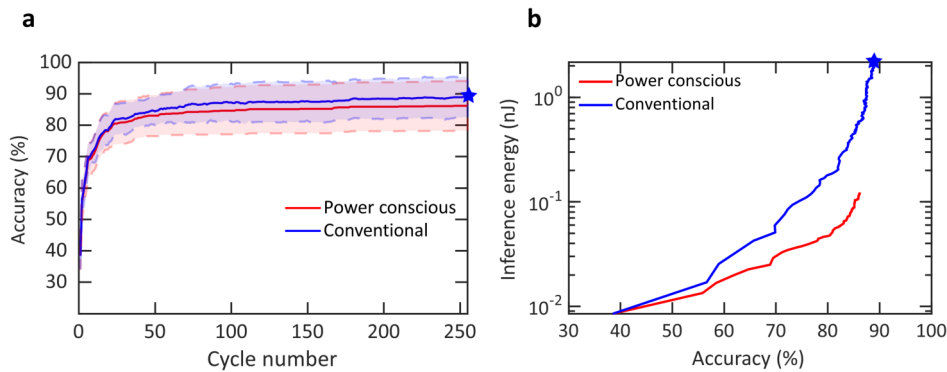


Figure 2.16: **Energy and recognition accuracy analysis of the Bayesian machine on a gesture recognition task.** **a** Mean accuracy according to the number of cycles in the inference for two types of computation: using a “power conscious” method by taking into account only the first one out for the decision (in red) and using the conventional stochastic computing by using the maximum number of one out for the decision (in blue). The shadows around the graph show one standard deviation of the mean accuracy over the ten subjects. **b** Energy consumption during the inference phase as a function of the accuracy for gesture recognition for the two methods. The stars correspond to the same point in both graphs **a** and **b**. All energy numbers are given for a supply voltage of 1.2 volts.

Before that, an obvious technique to lower the energy consumption is to reduce the number of cycles computed during stochastic inference. This reduction naturally impacts accuracy, as highlighted in Fig. 2.16a. This Figure shows, for the gesture recognition task, the accuracy of the Bayesian machine, as a function of the number of considered cycles. Numbers higher than 255 serve no purpose, as 255 is the periodicity of the eight-bits LFSRs. We consider the traditional stochastic computing strategy, as well as a “power-conscious” strategy. In the traditional approach, the system is operated for a fixed number of cycles, and the recognized gesture is chosen as the output that generated the highest number of ones. In the simplified power-conscious strategy, computation is stopped as soon as any of the circuit outputs produces a one, and this output gives the recognized gesture. We see that in both cases, cycle numbers as low as 50 allow approaching the accuracy obtained with 255 cycles. Based on these results, Fig. 2.16b shows the interplay between accuracy and energy consumption, using both strategies. The power-conscious approach consumes less energy than the conventional approach at equivalent accuracy. However, the power-conscious approach is limited to an accuracy of

86%, while the conventional approach can reach 89%. Overall reducing the number of cycles appears a highly effective strategy: in the conventional approach, accepting an accuracy reduction of only one percentage point allows reducing the energy consumption by a factor of 2.9.

To benchmark the energy efficiency of our approach, we also implemented the Bayesian gesture recognition task on a microcontroller unit (MCU), with an optimized approach using integer computation solely (see section Appendix section of this chapter). MCUs are tiny computers incorporating all their logic, volatile, and non-volatile memory on a single chip. They are currently the mainstream approach for providing AI at the edge in energy-constrained contexts [226]. Experimental measurements showed that recognizing one gesture with the MCU used 2.0 microjoules. Comparatively, the Bayesian machine, even when using 255 cycles, is using a total of 2.5 nanojoules to recognize one digit using the conventional approach, and 0.4 nanojoules using the power-conscious approach (with the maximum supply voltage of 1.2 volts). This is particularly impressive as our reference MCU is fabricated in a 90-nanometer CMOS node, comparable but more energy-efficient than the 130-nanometer CMOS node used for our Bayesian machine.

## 2.5 Nanodevice-Based True Random Number Generation

The energy efficiency of the stochastic Bayesian machine could significantly improve in the future. Fig. 2.15b reveals that in the current design, 75% of the energy is spent for the random number generation by the LFSRs and their distribution to the likelihood blocks (“vertical wires”). Several recent works have shown the possibility of generating random bits at a very low energy cost using stochastic nano-devices such as superparamagnetic tunnel junction (SMTJ) [18, 19, 216, 227] or random telegraph noise in RRAMs [228]. These proposals rely on natural fluctuations of the devices due to thermal or random telegraph noise and can therefore generate random bits relying only on read operation, at a very low cost. They could be distributed within likelihood arrays, thanks to their small area, and generate random bits at a much lower energy cost than LFSR (in the order of femtojoule/bit) and without requiring vertical wires. In the particular case of probabilistic bit, or “p-bits”, the Gupta circuit, which consumes in our design 22% of the inference energy could also be avoided, as this concept provides random bits with easily adjustable probabilities [18, 227].

In this section we introduce our work in collaboration with one of the leading spintronics research laboratories worldwide, SPINTEC, and CEA-Leti to design and fabricate several prototype circuits of True Random Number Generator and P-bits circuits based on the SMTJ devices and PCSA sensing circuitry.

### 2.5.1 Random Number Generation with MTJs

Random number generation refers to the process of generating a sequence of numbers that are intended to be random or unpredictable and do not follow a predictable pattern. The generation of high-quality random numbers is crucial for various applications, including not only probabilistic computing, but also cryptography, simulations, statistical sampling, and game programming.

In most cases, the generated numbers must be independent and decorrelated in time to ensure consistent results in mathematical operations. There are two main categories of random number generators: Pseudo-Random Number Generators (PRNGs) and True Random Number Generators (TRNGs). TRNGs are generated from physical sources that exhibit intrinsic entropy, such as thermal noise, while PRNGs are obtained through a deterministic system using a specific algorithm. TRNGs are considered more secure than Pseudo-Random Number Generators (PRNGs) as they are not based on deterministic algorithms and are therefore less susceptible to predictability and potential security flaws.

Various approaches and technologies have been used to generate true random numbers based on several types of entropy sources. Some works have proposed to generate random bits by exploiting the write operations of emerging memories (e.g., in spin-torque MRAM [229] or RRAM [230]). These proposals can generate high-quality random bits at high speed, but would

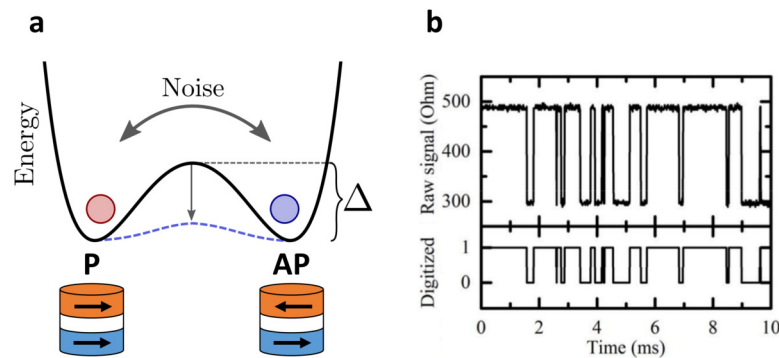


Figure 2.17: **Stochastic superparamagnetic tunnel junctions.** **a** Representation of the bistable magnetic states, and the associated low energy barrier (adapted from [18]). **b** Experimental resistance trace and thresholding operation (Reproduced from [19]).

not necessarily be adapted for the Bayesian machine. Due to the need for a write operation to generate a random bit, their energy consumption is comparable or higher than the LFSRs used in the current version of the machine.

Another interesting approach is to use the thermal noise as an entropy source, the unstable resistance state of a resistive device due to thermal noise is converted to a current or a voltage, and then processed by a comparator to compare the unstable voltage or current levels with the reference level to obtain a digital random signal. Some works have proposed to exploit the thermal noise effect on unstable spintronic nano-devices [18, 19, 216, 227], such as superparamagnetic tunnel junctions (SMTJs). SMTJs consist of a pinned nanomagnet and a free nanomagnet separated by a tunnel oxide layer. The free magnet can be in one of two states, parallel (P) or antiparallel (AP), with respect to the pinned magnet. The electrical resistance of the junction in the AP state is higher than that in the P state due to the tunnel magneto-resistance (TMR) effect.

The unstable SMTJ is designed with a low effective energy barrier between the two states compared to thermal noise, which causes the free magnet to spontaneously switch its magnetization direction between the two states, due to thermal noise (see Fig. 2.17a). This random switching can be observed as two-state fluctuations in the electrical resistance of the junction, which resembles a random telegraph signal. Resistance versus time measurements can be used to study the dynamics of these fluctuations (see Fig. 2.17b). These truly unpredictable resistance fluctuations can be converted to voltage or current fluctuations in the SMTJ device, then using a specific sensing circuitry those voltage or current fluctuations are converted to random digital bits.

To perform stochastic computation, such as in our Bayesian machine, we need a stream of random bits with a configurable probability distribution of 1s and 0s, to obtain a sum of 1s that is proportional to the binary value stored in the memory. For this purpose, our Bayesian ma-

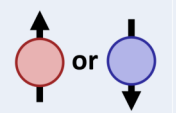
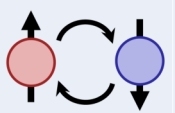
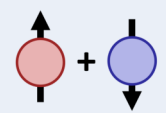
Computing paradigm	Digital	Probabilistic	Quantum
Basic unit	Bit	p-bit	q-bit
Hardware	Stable device	Unstable device	Single spin
State	Either 0 or 1	Fluctuation of 0 and 1	Superposition of 0 and 1
Illustration			

Figure 2.18: **Comparison table between conventional Digital, Probabilistic and Quantum computing paradigms.** Each column shows the specifications and a simple illustration of the computational paradigms, based on basic computational units: respectively, the bit, the p-bit and the qubit. (Adapted from [20]).

chine used LFSR for random number generation and GUPTA comparator for bitstream weighting. However, having one circuit based on nanodevices that can do both functions will lead to a considerable energy reduction in our system. The operation of generating a weighted (biased) probability of the random bit stream is related to the concept of probabilistic bit (p-bit). A probabilistic bit, or p-bit, is a two-level state system that can exhibit probabilistic behavior. Unlike classical bits, which can only be either 0 or 1, p-bits are represented by a distribution of 0s and 1s (see the table in Fig. 2.18).

Like TRNGs, a p-bit stream generation can be implemented through the use of magnetic tunnel junctions (MTJs). However, here a third terminal need is added to the sensing operation to bias the distribution of the outcome bitstream. The MTJs can be operated at room temperature, which is important for implementing them with classical computers.

### 2.5.2 Randomness Sensing With Precharge Sense Amplifier

True random number generators based on physical phenomena (such as thermal noise) are more desirable for stochastic computing but also challenging to implement with minimal energy consumption due to the energy cost of triggering random events. A hardware true random number generator typically consists of several consecutive steps to convert some aspect of the physical phenomena to a digital bit-stream. The simplified structure of a TRNG based on SMTJs consist of four main modules as shown in Fig. 2.19: (1) random signal is obtained from the unstable devices; (2) sensing and converting the random signal to digital signals; (3) if the obtained raw sequences does not satisfy a the uniform distribution, a feedback calibration signal is sent to the sensing circuitry; and (4) when a statistically-accepted sequence is obtained, the digital processing can be performed. Almost the same structure is applied for P-bits based

on SMTJs; the only difference is that the sensing circuitry has a biasing parameter to control the weighted probabilistic P-bit distribution.

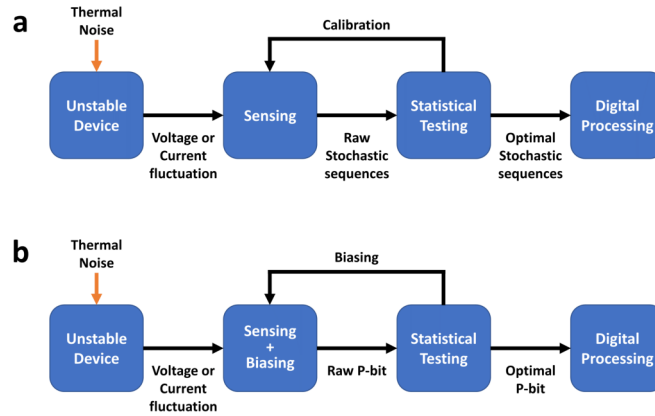


Figure 2.19: **Hardware structure for TRNG.** **a** Simplified hardware structure for TRNG with uniform distribution. **b** Simplified hardware structure for P bit generator. (Adapted from [21]).

Extracting random numbers that function by triggering random events (random programming of devices) or by amplifying the noise comes with large circuits and a non-negligible energy cost, making it less efficient. However, [19] proposed the use of a superparamagnetic tunnel junctions, which intrinsically amplify thermal noise without external energy supply. In the same work [19], the authors showed that SMTJ can generate high-quality random bits with minimal readout circuitry. In this work, and based on circuit simulations, the energy-efficient PCSA circuit was used to sense the outcomes (see Fig. 2.20a). The advantages of this approach are: no need for programming operation or programming circuitry for triggering random events, no need for circuits to amplify the noise, and use a low energy reading operation, and a fully compatible with standard CMOS fabrication processes.

In the work of [18], the authors proposed presented an Embedded MTJ-based p-bit; they show that the concept of a probabilistic bit can be implemented using a three-terminal circuit, based on a standard two-terminal MTJ with low barrier magnet (LBM) connected to the drain of NMOS transistors (see Fig. 2.20b). This structure is a voltage divider circuit with the MTJ resistance in series with a transistor resistance. The fluctuation of MTJ resistance (between  $R_p$  and  $R_{ap}$ ) will lead to drain voltage fluctuation  $V_m$ , to covert this fluctuation to digital bits, the  $V_m$  voltage is thresholded using an inverter. Based on SPICE models and simulation and using the transistor gate terminal, they have a tuned p-bit outcomes. While the concept certainly presents an intriguing and straightforward approach, it may benefit from a more comprehensive experimental exploration, particularly with regards to the read disturb effect. The observed steady current and voltages appear to be on the higher side, which might merit further investigation. In addition, the continuous application of sensing voltages, inherent to this concept, could potentially introduce concerns related to device longevity and increased energy



consumption. These considerations are shared with a view to enhancing the robustness and efficiency of the concept.

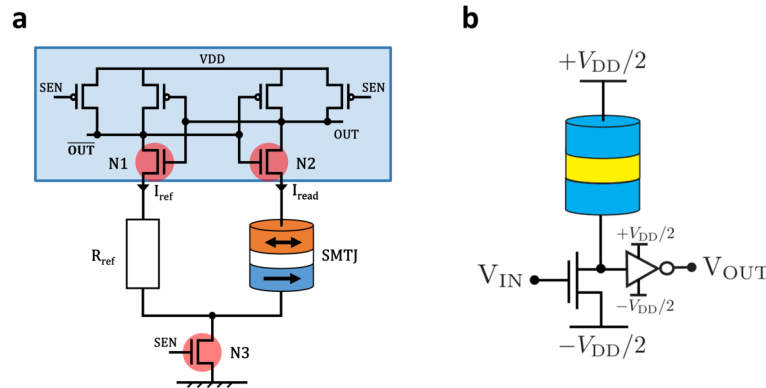


Figure 2.20: **Sensing circuitry for Stochastic magnetic tunnel junctions.** **a** PCSA based sensing circuit for reading the state of a SMTJ (Adapted from [19]). **b** a circuit for MTJ based P bit (Reproduced from [18]).

Finding the correct method and circuitry for sensing randomness is a crucial part of the TRNG process. Because many entropy sources are often quite fragile, applying inappropriate sensing current or voltages often accelerates the degradation of the quality of randomness, of even leads to failure to generate randomness. For this reason, continuous hardware or software statistical tests are suggested, but this leads to overwhelming circuitry and energy. The ideal solution is to design a sensing circuitry with minimum effects on the random phenomena and on the stochastic device.

Although most MTJ-based random number generator (RNG) approaches are compatible with conventional CMOS technology, it is rare to find experimental results based on fully silicon-fabricated solutions. Most works rely on SPICE simulations based on models derived from experimental data obtained from only MTJ device measurements.

In this work, we have addressed this issue by designing and fabricating hybrid MTJ-CMOS solutions for SMTJ-based RNG and SMTJ-based P-bit demonstrators. Our work encompasses the entire design process, including specification, circuit design, simulation, verification, and tape-out.

### 2.5.3 Design of SMTJ-Based RNG

In partnership with Spintec and CEA-Leti, we have collaborated on the design and the fabrication of a demonstrator chip for both True Random Number Generator (TRNG) and P-bit. These devices are based on SMTJs as the source of randomness and use, Precharge Sense Amplifier (PCSA), the same as circuit as the one used for reading memristors in the Bayesian machine (see section 2.2.3) as the sensing circuitry. Using PCSAs in this context was initially proposed in

a simulation study in [19].

The PCSA read scheme involves two phases: charge and discharge. This read scheme is particularly advantageous for fragile devices like SMTJs as it applies instant charge and discharge currents rather than continuous currents. Additionally, it maintains a low voltage difference between the device terminals, which increases the device's lifespan. However, it is crucial to limit the instant charge and discharge currents to avoid biasing the device's randomness.

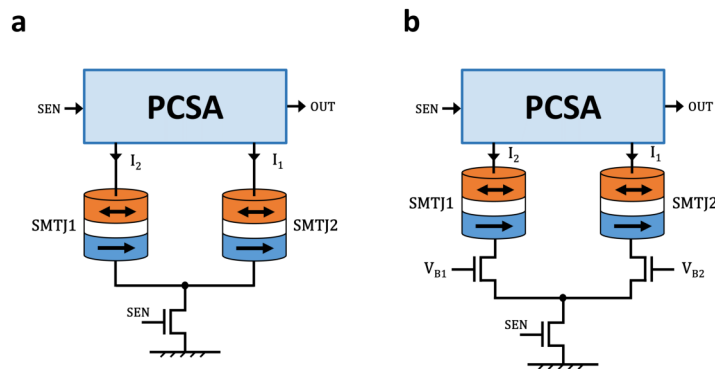


Figure 2.21: **Schematics of the designed TRNG and P-bit prototypes.** **a** Schematics of the first prototype TRNG circuit based on two SMTJs and a PCSA. **b** Schematics of the second prototype TRNG circuit based on two SMTJs, two biasing transistors and a PCSA, it can function as P-bit as well.

Our first design is a True Random Number Generator (TRNG) that produces a bitstream with an equal probability of 0s and 1s. Instead of using an unstable SMTJ in one side and a reference resistor in the other side like initially proposed in [19], we used unstable SMTJs on each side (see Fig. 2.21a). By using two sources of randomness, we can achieve a joint probability distribution, which increases the speed of fluctuations and is expected to improve the quality of the bitstream. The main challenge in this design was reducing the mismatch effect of transistors N1 and N2 (in Fig. 2.20a). Increasing their sizes led to relatively high instant current through the SMTJs, which is biasing the outcomes, thus limited the maximum size we can reach. Another idea was to reduce the size of transistor N3 (in Fig. 2.20a) or reduce its gate voltage to limit the maximum flowing current. However, this solution led to noisy discharging, which biased again the outcomes and affected the quality of the bitstream. We performed an investigation using SPICE simulations to find the best performance based on sizing compromises, and we validated the final design for tape-out and fabrication.

The investigation for the first design was based without taking into consideration the variability that occurs during the fabrication process of nano-devices. As devices shrink to the nanometer scale, process variations have a more pronounced impact on device performance and yield. Variability in the dimensions of magnetic tunnel junctions, for instance, can lead to changes in the device's resistance and switching behavior. In our case, device-to-device vari-

ability can cause a mismatch between the two SMTJ devices, resulting in a biased bitstream. To overcome this variability challenge, a second version of the design was developed, which added a biasing NMOS transistor in series with each SMTJ (see Fig. 2.21b). By controlling the gate voltage of these transistors ( $V_{b1}$  and  $V_{b2}$ ), we could calibrate the circuit to reduce mismatch between the two devices and control the maximum instant current through them.

In addition to reducing device variability, the biasing voltages also allow us to control the outcome probabilities by manipulating the  $V_{b1}$  and  $V_{b2}$  values to bias the bitstream towards one state more than the other. This additional functionality turns our design into a P-bit generator. Based on SPICE simulations, we selected the optimal transistor sizes and validated the design for tape-out and fabrication.

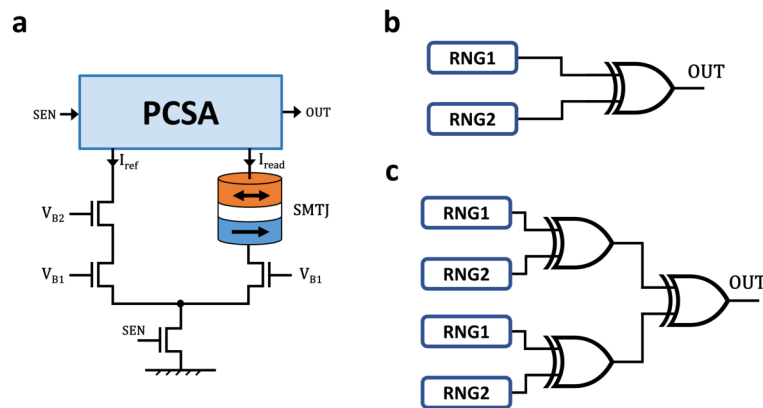


Figure 2.22: **Schematics of a designed P-bit prototype, and the XOR whitening technique.** **a** Schematics of the third prototype TRNG circuit based on two SMTJs, two biasing transistors and a PCSA, it can function as P-bit as well. **b** Schematics of the XOR2 whitening circuit using two basic RNG circuits and one XOR2 circuit. **c** Schematics of the XOR4 whitening circuit using four basic RNG circuits and three XOR2 circuits.

Our third design was inspired by a concept presented in [19] that been presented in Fig. 2.20a, which uses an unstable SMTJ on one side and a reference resistive device on the other side. However, in our design, we chose to use a transistor as the reference resistive device (see Fig. 2.22a). By controlling the gate voltage  $V_{b2}$  of this transistor (N5), we can calibrate the circuit for TRNG or manipulate the outcome probabilities to turn it into a P-bit generator.

Similar to our second design, we also incorporated the two transistors to control the maximum instant current through the SMTJ. Using SPICE simulations, we optimized the transistor sizes and validated the design for tape-out and fabrication.

In case our basic designs could not generate random bits with high enough quality, a “whitening” technique can be applied on the raw random bit outcomes. The whitening process involves transforming a set of raw or biased random bits into a more uniform and unbiased set of random bits. This is typically achieved by combining multiple independent bitstreams using

Type of Cell	RNG	P-bit or RNG	P-bit or RNG
Basic designs	PCSA 2SMTJ	PCSA 2SMTJ-VB	PCSA 1T1SMTJ
Declinations	XOR2 2SMTJ	XOR2 2SMTJ-VB    XOR4 2SMTJ-VB	XOR2 1T1SMTJ

Figure 2.23: **Classification table of the designed prototypes.** The table lists and classify the designs . First row for the basic designs and second row for the designs with the XOR whitening technique. and Based on the type of the design. One column per prototype.

XOR gates. The XOR2, XOR4 and XOR8 whitening methods are methods that combines respectively two, four and eight independent bitstreams (from RNG) using XOR gates (see Fig. 2.22b).

This technique reduces auto-correlation exponentially and brings the mean state closer to a perfect balance. The more bitstreams combined, the greater the reduction in auto-correlation and bias, and the more the sampling frequency is increased.

Whitening can eliminate biases in the raw random bit outcomes of our three basic RNG designs using off-chip (software or hardware XOR) or on-chip XOR circuitry. Although off-chip XOR is easier to implement, it is less accurate in terms of energy estimation. On-chip XOR implementation allows for more precise real-world experiments and energy measurements. We implemented XOR2 and XOR4 whitening circuits based on our three RNG designs. We created new derivative designs with XOR2 whitening circuits based on design one and design three, and new derivative designs with both XOR2 and XOR4 whitening circuits based on design one and design three (see table in Fig. 2.23).

We have designed seven prototyping circuits, the designs tested by simulations than are taped out for fabrication. Because we use 25 input/output pads designed for characterization by a custom probe card, we needed to set of pads to route all of our designs (see Fig. 2.24a).

The tapeout of our designs was fabricated in a hybrid CMOS/SMTJ process (see Fig. 2.24b). The CMOS part of the circuit is fabricated using a low-power foundry 130-nanometer process with four layers of metals. SMTJ devices are fabricated on top of the CMOS foundry layers by our partners in Spintec and CEA-Leti.

At the time of writing this thesis, we are expecting the delivery of the fabricated dies any day.

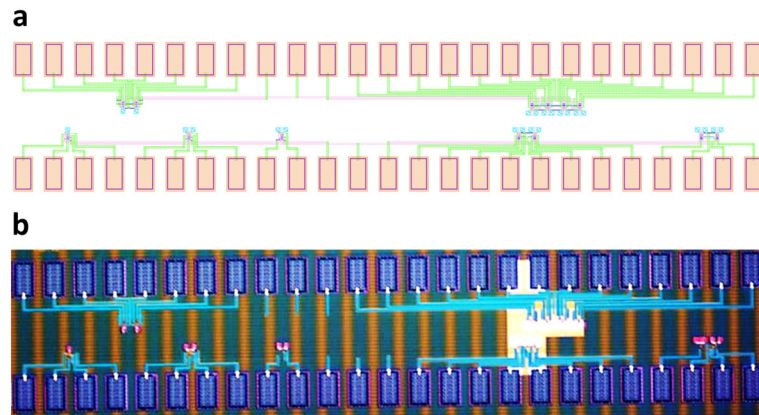


Figure 2.24: **The SMTJ Based RNG and P-bit demonstrator chip.** **a** Layout view of the chip with the seven designs, using two sets of 25 IO pads. **b** Optical microscopy photograph of the fabricated die.

## 2.6 Conclusion

In this chapter, we have shown that a Bayesian machine can be implemented in a system with distributed memristors, performing computation locally and with minimal energy movement. This allows it to perform Bayesian inference with an energy efficiency orders of magnitude higher than that of a standard microcontroller unit. Due to its reliance on non-volatile memory, and its sole use of read operations once the likelihoods have been programmed, the system can be powered down while regaining functionality instantly. It can also be operated at low, and possibly varying, supply voltages. While Bayesian models are usually considered computationally expensive, our results suggest that complex models could be embeddable at the edge, with low power consumption. This could allow edge systems to benefit from the qualities of Bayesian inference to deal with highly uncertain situations with little data, and to make predictions using an explainable mode.

Explainability is desirable in many critical situations for ethical and regulatory reasons[77]. The fact that Bayesian inference takes decisions based on explainable models also has practical consequences, in particular when we use them with inputs that differ from those used for training the model: the Bayesian machine excels at recognizing situations where it cannot provide a reliable answer. For example, Supplementary Note 10 of our published work [14] shows that when we present a gesture from a different subject than the one for which it has been trained, the Bayesian machine provides a clear signature that it cannot provide a certain output. This feature could be particularly useful for medical devices, where wrong decisions can have dramatic consequences. Supplementary Note 11 of our published work [14] also illustrates the possibility to train the Bayesian machine with little data: a mean accuracy of 78% can be obtained on gesture recognition using only two examples per gesture. This possibility originates in the natural capacity of Bayesian models to generalize and also in the possibility to

incorporating prior assumptions on the model into the likelihood training process.

The design choices of the Bayesian machine were led by the specificities of Bayesian inference. The need for eight-bit likelihood, a precision higher than what analogue memristors can provide, and the fact that Bayes' law does not require multiply-and-accumulate, led to a digital design. In contrast, most memristor-based neural network accelerators use analogue computation, at least to some extent. This allowed us to rely on a simple sense amplifier for reading memristors, which brings multiple advantages: it is highly flexible in terms of supply voltage, functions without needing any calibration, mitigates read disturb, and is largely immune to device variation. The simplicity of the sense amplifier also allowed us to demonstrate a complete system featuring 16 small memory blocks, whereas analogue memristor-based neural networks usually have a single memory block.

In addition, a benefit of the use of stochastic computing by the Bayesian machine is that our system is naturally resilient to soft errors: bit errors can make one cycle wrong, but will be averaged throughout the computation. (This point is illustrated in Supplementary Note 9 of our published work [14], which demonstrates the resilience of the gesture recognition tasks to single-event upsets.) As memristor storage is also more resilient to radiation than static RAM [231], this feature can make the Bayesian machine appropriate for extreme environments. All these features make the Bayesian machine robust and flexible, and it can therefore be particularly useful for monitoring difficult environments with variable or unstable power supply. This capability maps well with the fact that Bayesian excels at dealing with the highly uncertain situations encountered in such environments (see Supplementary Notes 10 and 11 of our published work [14]).

The results achieved in this study, as shown by the measurements performed on the demonstrator chip and the energy estimates based on the scaled-up design, have encouraged us to take the project further. Specifically, by creating a system that has both larger memory and higher computing capacity, which will allow us to implement real tasks on the chip. To achieve this goal, a modified version of the scaled-up system was designed and sent for fabrication (see Section 3.5).

Our research utilized a 130-nanometer process, demonstrating that inexpensive technology can achieve energy efficiency. Since digital circuitry dominates energy consumption, scaling the design to more advanced technology nodes can further reduce energy consumption. Our energy analysis indicated that during the inference phase, 88% of energy consumption resulted from random number generation and distribution. The cost of generation is due to the use of LFSRs, while the non-local nature of random number generation leads to distribution cost. Our system employed a single 8 bits LFSR per column, which was shared by all the likelihood blocks of the column.

The energy of random number generation could be reduced by again using nanodevices (see section 2.5), based on the study in [19] about using stochastic nanodevices to generate high-quality random bits locally, at a very low area and energy cost, using read operations, we

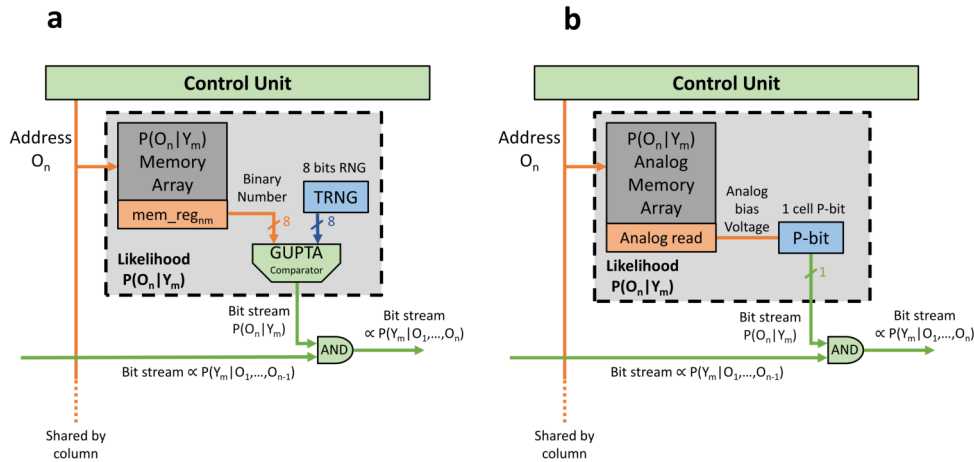


Figure 2.25: **Suggested integration of TRNG and P-bit to the Bayesian machine.** **a** Schematics of likelihood with embedded TRNG. **b** Schematics of likelihood with embedded P-bit, the likelihoods are stored in analog values, the analog read of memory outputs analog biasing voltages corresponding to the stored probability.

have designed and fabricated several prototyping circuits, that uses unstable SMTJs and PCSA sensing circuitry for generation random bits (RNG or P-bit).

Anticipating future experimental results and future advancements in device integration, we propose several ideas for integrating our circuits into the Bayesian machine to reduce energy costs. One idea is to replace the 8-bit LFSR circuit with an 8-bit TRNG circuit. While this will reduce the energy cost of the LFSR, the energy of the GUPTA circuit and distribution will still remain. Another idea is to embed an 8-bit TRNG circuit as shown in Fig. 2.25a, which will reduce distribution costs at the expense of increasing the total number of TRNG circuits.

If we completely change the design of the Bayesian machine by adopting analog memory for storing likelihoods, we could embed a P-bit circuit with the analog biasing voltage for each likelihood, as shown in Fig. 2.25b. The probability value is read from the analog memory, converted to analog voltage that will bias the P-bit to generate random numbers proportional to the read probability. This approach improves memory density because of using the analog approach and eliminates the need for energy-hungry ADCs. Additionally, only one P-bit cell is required instead of 8 TRNGs, and eliminating the need for the GUPTA circuit. While this approach offers many advantages, challenges remain due to the lack of technological maturity of the devices, high noise and variability problems with analog storage, and the ongoing exploration of the behavior of unstable SMTJs.

## Appendix: Methods for the Evaluation of the Scaled-Up Bayesian Machine (section 2.4)

### Gesture recognition task on a Bayesian machines

The gesture recognition task is realized on a dataset collected in-lab, including ten subjects. Each subject was asked to perform four gestures (writing the digits one, two, three, and a signature specific to each person) in the air. The subjects were not given any instruction on how to perform the gestures, leading to a high diversity within the dataset. The gestures were recorded using the three-axis accelerometer of a standard inertial measurement unit. Each subject repeated the same move between 25 to 27 times. The recording time varied by subject and gesture, and ranged from 1.3 to 3 seconds. We extracted ten features, named  $F_0$  to  $F_9$ , from each recording, after filtering of the gravity: mean acceleration, maximum acceleration on the three axis, variance of the acceleration on the three axis, mean value of the jerk of the acceleration, and maximum value of the jerk of the acceleration on the three axis.

We train the system using 20 of the 25-27 recordings for each subject, and the last 5-7 recordings are used to test it. Our model assumes a uniform prior, and the resulting machine, therefore, features no prior block. Training consists in adjusting the likelihoods  $p(O_n|Y = y)$  to the training data. As the training data is very limited, we fitted these likelihoods by Gaussian functions (using the `fitdist` MathWorks MATLAB function). We then discretized the resulting Gaussian distributions to the 512 possible input values of the observations in the scaled-up Bayesian machine. We then normalize the probabilities in each column of the Bayesian machine so that the maximum value is one. Finally, we quantified the normalized probability values to eight-bit integers, with the value zero equivalent to  $1/256$  and 255 to  $256/256$ .

To optimize further the energy consumption of the system, we use only six of the extracted ten features in the Bayesian machine. Based on a systematic study, the features deleted for the experiment are  $F_0, F_2, F_3, F_8$ . Additionally, we realized that broadening the Gaussians obtained when fitting the data allowed stochastic computing to converge faster, allowing the system to reach better accuracy. Therefore, in all our results, the standard deviation of the Gaussian in the fitted likelihoods is multiplied by a broadening coefficient of 1.3 with regards to the initial fit.

In the conventional technique, the answer of the Bayesian machine to an output is given by the argmax of the number of ones outputted by each row. In the power-conscious method, the answer is the row that produced the first one. In both cases, if no output produced a one, the answer is considered an error.



### 2.6.1 Energy Estimation of a scaled stochastic Bayesian machine

In order to implement the Gesture recognition task, the memory capacity of the fabricated Bayesian machine was not enough. A scaled-up Bayesian machine need to be designed. The scaled-up machine features six columns and four rows of likelihood blocks. Each likelihood block features an array of  $128 \times 64$  memristors arranged using the same differential structures as our test chip, therefore implementing four kilobits per array. We developed a behavioral MathWorks MATLAB model of the machine, a synthesizable SystemVerilog description, and test benches for both models using consistent input files. Both models were verified to be equivalent for all possible inputs and for all cycles. We synthesized the SystemVerilog description and placed and routed the whole Bayesian machine in our reference technology (see Fig. 2.14b). Post-place-and-route simulation, including the delays due to the gates and the parasitic capacitances, gave results that still matched perfectly the MATLAB model of the Bayesian machine.

The energy estimation of the Gesture recognition task on the scaled Bayesian machines was performed using a homemade framework using a hybrid methodology. These estimates focus on the inference phase, i.e., the actual operation of the Bayesian machine when various inputs are presented, after the memristors have been formed and the likelihoods programmed. The energy consumptions of the memristor arrays themselves are obtained using circuit simulations (based on the Siemens Eldo simulator), including the parasitic capacitance extracted from the memory array layout. The energy consumption of the rest of the system is obtained using the Cadence Voltus power integrity solution framework. These estimates use value change dump (VCD) files obtained from our test bench, ensuring that the energy estimates correspond to a realistic situation.

A challenge of these estimates is that when estimating energy consumption, it is crucial that Cadence Voltus models the behavior of the memristor arrays properly, as the energy consumption of the system depends directly on the output of these blocks. However, as they are custom blocks, and not included in the standard library of the foundry, special developments were required. We programmed, using MATLAB, a memristor array to liberty file compiler, providing, based on the likelihoods programmed in a memory block, a file describing to Cadence Voltus the functionality of the array. During the MATLAB simulation, we extract the intermediates values of the inference, and we include them as a memory output. For that purpose, we create a new liberty file that will be used in the place-and-route operation. This liberty file specifies the output for each memory as a function of the input and addresses. This method can only be used to estimate the energy consumption during the inference phase as the outputs are tailor-made for this phase.

As a benchmark, we also implemented the gesture recognition task on an ST Microelectronics STM32F746ZGT6 microcontroller unit (MCU, integrated on a test Nucleo-F746ZG board). This type of MCU, manufactured in a 90-nanometer CMOS process, is commonly used for edge artificial intelligence. Our implementation was programmed in the C language using the

---

ST Microelectronics STM32 Cube integrated development environment and was optimized for running on the MCU (the stochastic computation of our test chip is replaced by standard integer addition, and the likelihoods were replaced by log-likelihoods to avoid multiplications). To perform the benchmark, the MCU computes gesture recognition for all possible inputs sequentially and blinks an LED on the board every one-million inference to allow precise timing. The energy consumption of the MCU was measured using a standard Ampere meter (the energy consumption of solely the MCU was measured, excluding all other components on the board). To isolate the energy consumption strictly due to Bayesian inference, we also measured the energy consumption of a control program that includes all operations performed by the gesture recognition program (looping inputs, LED blinking...), except the actual Bayesian inference. Then, we subtract the energy consumption of the full gesture recognition program ( $2.4\mu J/\text{gesture}$ ) and the one of the control program ( $0.4\mu J/\text{gesture}$ ) to get the energy used by the MCU strictly for Bayesian inference ( $2.0\mu J/\text{gesture}$ ).



## Chapter 3

# A Logarithmic Bayesian Machine

“There’s a Way To Do It Better — Find It”

---

Thomas EDISON

Mathematics is a cornerstone of engineering, with a plethora of techniques available for modeling, analyzing, and solving complex problems. By altering the computational domain, a range of benefits can be obtained, such as improved security and efficiency in cryptography by employing different number fields, optimization of control systems by altering matrix space or eigenvalue, signal processing by analyzing in the frequency domain, and reduced power consumption through the use of probabilistic or logarithmic representations.

To address the area and energy-intensive nature of classical computation circuitry in memristor-based Bayesian machines, stochastic computing has been successfully employed, as outlined in the previous chapter, with minimal area requirements and low energy consumption. However, stochastic machines suffer from limitations in precision, inference speed, and complexity with random number generation. In response, in this chapter, we turn to the logarithmic computing approach, which offers increased precision and faster inference operations.

Logarithmic computing is particularly suitable for Bayesian inference, as it allows computing the product of the prior distribution and the likelihoods using simple addition and subtraction operations. This makes it an ideal choice for hardware implementation, where speed and efficiency are crucial factors. Logarithmic probability representation also provides an advantage over traditional arithmetic, preventing numerical underflow and loss of precision when dealing with small probabilities or large datasets.

This chapter examines the use of logarithmic computing in our Bayesian machine architecture, utilizing the same design choices as the stochastic version, such as memristors and near-memory computing architecture. We demonstrate how logarithmic computing can be implemented and why it is poised to improve the energy efficiency and accuracy of Bayesian inference. Moreover, we present a newly fabricated logarithmic Bayesian machine integrated circuit, which has undergone recent testing. On-chip measurements on both machines, the logarithmic and the stochastic, reveal the viability of our Bayesian machine approach, even in the presence of memristor imperfections. Our Bayesian machines can operate at low supply voltages, and scaled-up versions are capable of performing a gesture recognition task using orders of magnitude less energy than a microcontroller unit. Additionally, we provide the first explicit comparison of stochastic and logarithmic computing in a near-memory computing integrated circuit with nanodevices, comparing their energy efficiency.

The results in this chapter are adapted from an article presented at the DATE 2023 conference [86].

## 3.1 Bayesian Inference with Logarithmic computing

The invention of logarithms by John Napier in 1614 revolutionized the way that calculations were performed, making them faster and easier. Although calculators have replaced humans for calculation, the concept of simplifying calculations is continued to be used in the modern era of calculators and computers in the context of emerging non-conventional computing paradigms [104, 168, 232], due to the need for area and energy-efficient computation [233–236] for wearable and energy-constrained IoT devices [237]. The use of logarithms has once again been employed to simplify and streamline computations [209, 210], specially for improving the energy efficiency of the artificial intelligence edge systems.

As most of the research on energy-efficient AI systems has focused on deep neural networks (DNNs), logarithmic computing has already been used to improve the efficiency of those neural networks [238–242]. The logarithmic number representation has been employed in convolutional neural networks in the work of [238]. Based on the results in this work, the logarithmic approach can handle non-uniform distributions of weights and activations, and enables state-of-the-art networks to be encoded in an extremely reduced bits representation with negligible loss in classification performance, improving on fixed-point representations. For some specific applications, logarithmic data representation is more robust to quantization than fixed-point, eliminates bulky digital multipliers, and reduces memory requirements, area, and energy consumption. Similarly, [239] proposed using logarithmic encoding of non-uniformly distributed weights and activations to reduce power consumption and increase inference speed. The authors of this work demonstrated their ideas in LogNet, an inference engine using only bitshift-add convolutions and weights distributed across the computing fabric.

Although logarithmic arithmetic simplifies multiplication and division, it can render additions and subtractions more complex, which poses challenges for deep neural networks that depend heavily on multiply and accumulate (MAC) operations. However, Bayesian inference is particularly well-suited for the logarithmic computing approach as it relies on computing the product of the prior distribution and the likelihoods, followed by normalizing the result to obtain the posterior distribution. With logarithmic arithmetic, Bayesian inference requires only simple addition and subtraction operations, making it easier to implement in hardware. This makes it a preferred option for implementing Bayesian inference in hardware, where speed and efficiency are paramount. Furthermore, the use of the logarithmic representation for probabilities offers another advantage over traditional arithmetic. Specifically, it prevents numerical underflow and loss of precision when dealing with small probabilities or large datasets, which can occur when using low-precision fixed-point representation.

In the stochastic Bayesian design (explained in Chapter 2), the probability values of the likelihood range between 0 and 1 and are represented with an unsigned eight-bit fixed-point format, which quantizes only the fractional part in the linear scale, resulting in a loss of precision for small probability values due to fixed step-size between all representation levels. A

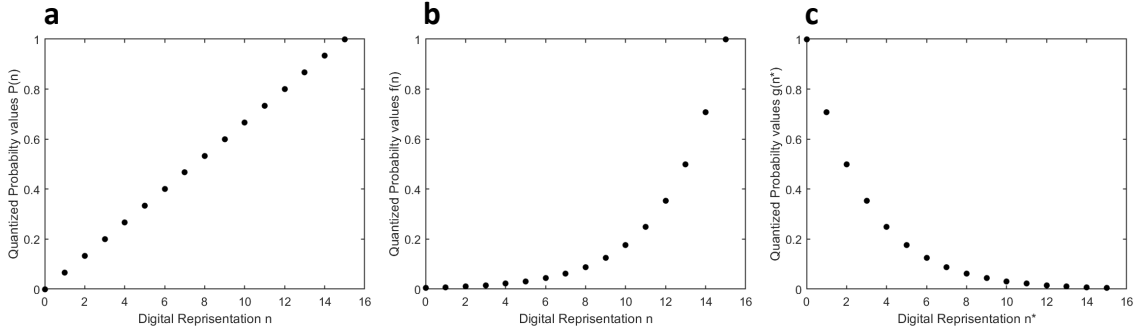


Figure 3.1: **Quantization functions.** **a** Linear quantization of probabilities represented by an unsigned four-bit fixed point format. **b** Logarithmic quantization of probabilities represented by an unsigned four-bit fixed point format. **c** Logarithmic quantization of probabilities represented by an unsigned one's complement four-bit fixed point format.

four-bit fixed-point representation, for instance (see Fig 3.1a), can only represent probability values down to 0.0625, leading to a loss of precision for smaller probability values.

A logarithmic representation with log-scale quantization can overcome this limitation by assigning more code values to smaller probability values on a logarithmic scale, resulting in an increased resolution in lower probability regions. To apply this representation, the number of bits ( $N$ ) used to store the probability and the logarithmic base ( $B$ ) must be determined. As the base represents probabilities, it must fall within the range of  $0 < B < 1$  and be approximated with respect to the logarithmic quantization function  $f(n)$ :

$$f(n) = B^{(K-n)/m}. \quad (3.1)$$

The binary probability value of  $f(n)$  is stored as  $n$ , where  $K = 2^n - 1$  is the maximum number in the digital representation. To maximize the system's accuracy for specific use cases, the number of quantized probabilities  $m$  that satisfy  $f(n) > f(m) = B$  is determined. For example, Fig 3.1b shows a function  $f(n)$  used with 4-bits numbers,  $B = 1/2$  as the logarithmic base, and  $m = 2$ . This configuration allows 14 binary values to represent probabilities between  $1/2$  and 0, with the smallest possible probability value of  $f(0) = 0.0055$  (close to the eighth-bit fixed-point resolution of 0.0039), and only two binary values for probabilities between 1 and  $1/2$ .

Although the function  $f(n)$  in equation 3.1 represents an increasing function where binary values increase with probability values, it is not practical for converting multiplication to addition. For example, adding the binary numbers  $n_1 = 1$  and  $n_2 = 2$  results in  $n_3 = 3$ , but the product of  $f(n_1) \times f(n_2) \neq f(n_3)$ . This is because  $f(1) \times f(2) = B^{(K-1)/m} \times B^{(K-2)/m} = B^{(2K-3)/m} \neq B^{(K-3)/m}$ . Thus, the product cannot be implemented by a simple adder. One solution to this problem is to use the one's complement representation  $n^*$  of the digital number  $n$  (without the use of sign bit), where  $K = 2^n - 1 = n + n^*$ . By replacing  $K - n$  with  $n^*$  in equation 3.1, we obtain the new quantization function  $g(n^*)$ :

$$g(n^*) = B^{n^*/m}. \tag{3.2}$$

The function  $g(n^*)$  in eq 3.2 represents probability as a decreasing function, where the probability values decrease with increasing binary values. For instance, Fig 3.1c displays a function  $g(n^*)$  with  $N = 4$  bits, base  $B = 1/2$ , and  $m = 2$ , where the smallest possible probability value is represented by the biggest binary value 1111, which corresponds to the probability value  $g(15) = 0.0055$ . Using the new function, adding the binary numbers  $n_1^* = 1$  and  $n_2^* = 2$  results in  $n_3^* = 3$ , and the product of  $g(n_1^*) \times g(n_2^*) = g(n_3^*)$ , as explained by  $g(1) \times g(2) = B^{1/m} \times B^{2/m} = B^{3/m} = g(n_3^*)$ . This means that the product can be implemented using a simple fixed-point adder.

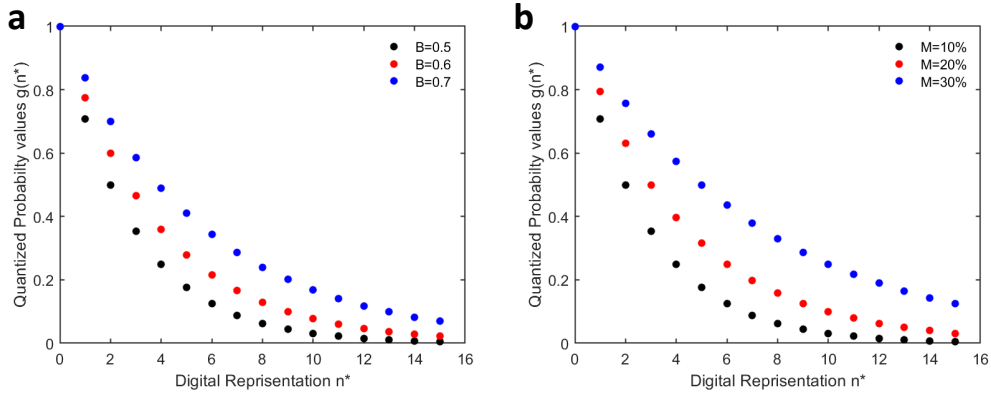


Figure 3.2: **Determining the Logarithmic Quantization Parameters.** This Figure explores the impact of two parameters on quantized probabilities: (a) the logarithmic base and (b) the m parameter. Increasing either parameter leads to a better representation of high probability values at the cost of reduced precision and increased minimum encoded probability value.

The function  $g(n^*)$  has two parameters,  $B$  and  $m$ , which need to be determined for better approximation and precision of encoded probabilities. In Fig 3.2a, the effect of the base  $B$  was studied, with  $m$  fixed at 2 and three chosen base values of  $B = 0.5, 0.6, 0.7$ . As illustrated in Fig 3.2a, the base  $B$  has an impact on the quantized probabilities. For higher base values, the quantization function becomes more linear. On the other hand, it also affects the minimum encoded value, which increases with the increasing base values (0.0055 for  $B = 0.5$ , 0.021 for  $B = 0.6$ , and 0.069 for  $B = 0.7$ ), leading to a loss of precision for low probabilities.

Based on the parameter  $m$ , we create another parameter  $M = \text{Round}(m/2^N)$ , which represents the percentage of encoded probability values with  $g(n^*) > g(m)$  over the number of represented levels. In Fig. 3.2b, the effect of choosing  $M$  was studied, with  $B$  fixed at 1/2 and three chosen base values  $M = 5\%, 10\%, 20\%$ . As illustrated in Fig. 3.2b, by increasing  $M$ , we achieve better approximation for probabilities higher than 1/2. However, it gradually increases the minimum encoded probability. For example, with  $M = 30\% (m = 5)$ , the minimum encoded probability is  $g(15) = 0.125$ , which means that all probabilities less than that minimum are rep-



resented with the same value, leading to an increase in rounding errors.

The determination of the logarithmic quantization parameters depends on the application, the computation model, and the data distribution. In the case of the Bayesian machine, it is recommended to set this value in such a way as to have more probabilities between  $1/2$  and  $0$  than between  $1$  and  $1/2$ . The main reason for this is that some events have quite low probabilities of occurring, and therefore, to increase the accuracy of the system, a low value is needed. Therefore, in a Bayesian system, probabilities between  $1/2$  and  $0$  are more important than those between  $1$  and  $1/2$ . While the logarithmic quantization method does not allow us to encode the exact probability of an event, it provides a reliable approximation of the most probable event, this approximation is made in such a way as to minimize the risk of errors.

### 3.2 Design of a Logarithmic Bayesian Machine

The successful implementation of the memristor-based Bayesian machine with a stochastic computing approach has prompted us to utilize the same architecture and design choices for the logarithmic computing approach. As a reminder, Bayesian inference aims to evaluate the probability of an event  $Y$  based on a collection of observations  $O_1, \dots, O_n$ , using Bayes' law [188, 189]. If all observations are conditionally independent, Bayes' law simplifies to the product of prior probability  $p(Y)$  and likelihood factors  $p(O_i|Y)$ , known as the naive Bayes' law as presented in Chapter 1 (see Equation 1.4). By applying a logarithmic function with a base  $B$  to the naive Bayesian equation, we obtain the logarithmic form of the naive Bayesian equation in 3.3:

$$\log_B p(Y|O_1, \dots, O_n) \propto \log_B p(O_1|Y) + \dots + \log_B p(O_n|Y) + \log_B p(Y). \tag{3.3}$$

The logarithmic form of Bayes' law simplifies the product of prior probability and likelihood factors to the sum of logarithmic likelihoods. The idea of applying the logarithm to the likelihoods (log-likelihood) is already been used in the maximum of likelihood method for parameters estimation (used for fitting the likelihood distributions). This is because the logarithmic function is monotonically increasing, ensuring that the maximum value of the logarithm of the probability occurs at the same point as the original probability function. The decision-making in our Bayesian machine relies on the maximum of the posterior probabilities of an event, making our machine unaffected by the logarithmic transformation of the probabilities. Therefore, we can implement the simpler log-likelihoods in our Bayesian machine instead of the original likelihoods.

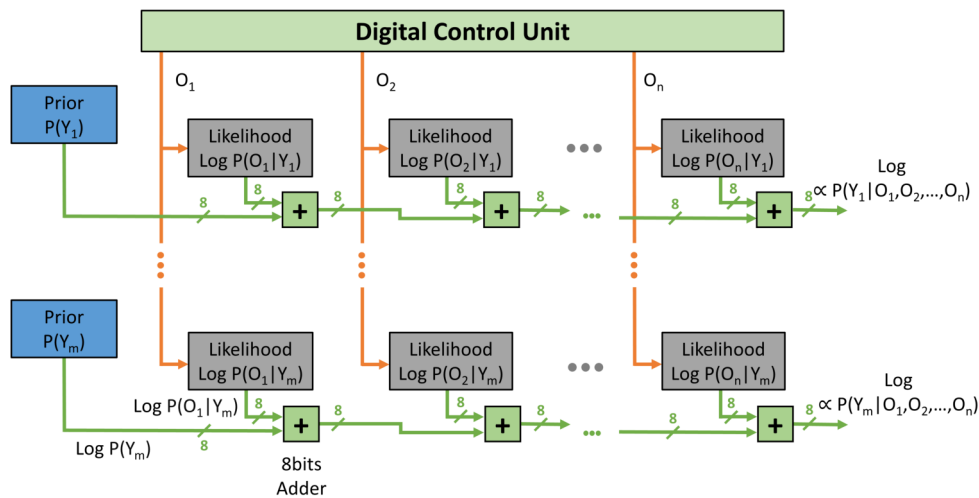


Figure 3.3: General architecture of the Logarithmic Bayesian machine.

Our logarithmic representation has the base  $B = 1/2$  and  $m = 32$  ( $M = 12.5\%$ ), and it is approximated with respect to the logarithmic quantization function  $g(n^*)$  in Equation 3.2. Because this function represents probability as a decreasing function, where highest probability value are represented by lowest binary values, the decision-making in our Bayesian machine relies on finding the minimum of the digital values.

The memristor-based Bayesian machines architecture for logarithmic design is similar to the one used for the stochastic design (see Fig 3.3): it is obtained by implementing equation 3.3 topologically, using near memory computing paradigm. Each likelihood factor is stored in an independent memory array using the logarithmic representation of eight-bit, and computations are performed physically near these memory arrays. The computation result is then passed to the following memory array. The observations  $O_1, \dots, O_n$  act as addresses for the memory arrays, telling the likelihood value to be read. The concept of distributed near-memory computation allows the circuit to function with minimal energy consumption due to minimal data movement.

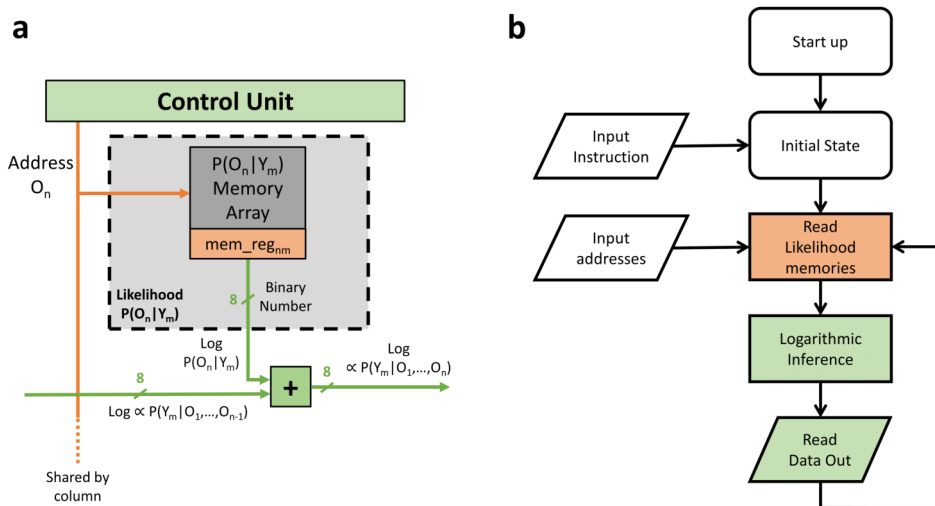


Figure 3.4: **Detailed operation of the logarithmic Bayesian machine.** **a** Schematic illustrating the detailed architecture of a Log-likelihood elementary block. **b** Flowchart of the different operations to perform a Bayesian inference computation in the logarithmic Bayesian machine.

In the logarithmic design, the probabilities are stored in the memory arrays in the logarithmic domain. Fig. 3.4a detailed schematics of the log-likelihood elementary block, which has one input vector representing the observations  $O_n$  in form of memory addresses of probability values, and an eight-bits output vector. The log-likelihood block consist of: an  $8 \times 8$  memristor array of  $2T2R$  structure, including all read and program circuitry, and an eight-bits unsigned fixed-point adder. Because addition is limited to eight bits, the adder was designed with an

overflow detect and saturate output: if the adder detects overflow, the output is saturated to the maximum digital output "11111111" (or  $n^* = 255$ ). This means that the maximum digital output represents the minimum encoded probability  $p(255) = 0.004$  (the same as minimum encoded by unsigned 8 bits fixed point with linear quantization).

Similar to the stochastic design, we use a two-transistor-two-resistor (2T2R) strategy in which memristors are used in a binary fashion. We also adopt the same memristor read and write strategies explained in Chapter 2.

Fig. 3.4c shows a time diagram of the machine operation. The color code throughout Fig. 3.4 is as follows: the orange color is used for memory read, and the green color for the actual logarithmic inference. The inference operation in the logarithmic Bayesian machine is much easier than the stochastic one, the inference performed in three main phases:

- **Memory read.** Likelihood values are read from the memristor arrays, based on the input observations (acting as row addresses). Observations  $O_1, \dots, O_n$  are off-chip inputs loaded from dedicated input pads, then addressed to the likelihood memory arrays by the Bayesian machine digital control unit (one observation for each column). All likelihood memory arrays are read simultaneously.
- **Logarithmic Bayesian inference phase.** In one clock cycle, all addition operations are performed; the results of those additions represent the logarithmic Bayesian inference outputs. Those outputs are sent to the read-out shift registers.
- **Read Data Out.** Inference Outputs are stored in read-out shift registers with parallel inputs and serial outputs. Each row of the logarithmic Bayesian machine has an eight-bit output. Due to the limited number of IO pads, we needed an eight-bit shift register for each row, to convert the parallel eight-bit outputs to one-bit serial outputs. Therefore, it takes eight clock cycles for the machine to read the outputs.

The design process for the logarithmic Bayesian machine was accomplished with a significantly reduced time and effort compared to the stochastic version, thanks to the implementation of a homemade complete automated place and route design flow, as described in Chapter 1. This flow automates the placement and routing of memory blocks, in addition to digital logic standard cells. The memory block is considered a standard cell in this flow, and as such, it has a Liberty Timing File (.lib) that represents the timing and power parameters associated with the memory cell. Furthermore, an abstract view of the memory cell shown in Fig. 3.5b is generated based on the layout view of the memory block, as shown in Fig. 3.5a. This abstract view is crucial for the automatic placement and routing of the memory block to the multi-supply voltages (see Fig. 3.5c) and to other standard cells.

To ensure the reliability of the new design flow, all physical verification of the final design was conducted using dedicated Calibre EDA tools. This included design rule checks, layout-versus-schematic comparison, and antenna effects design rule checks. Still, it is imperative to

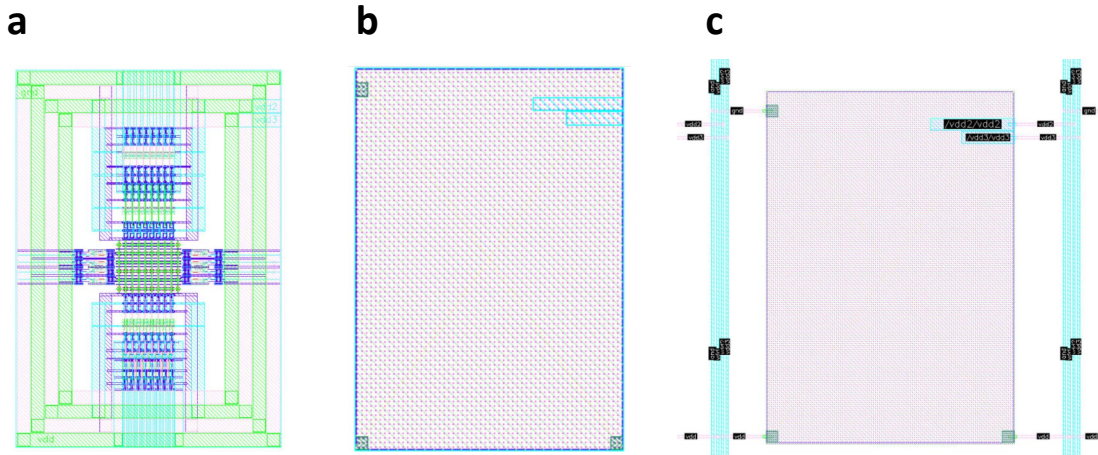


Figure 3.5: **Physical views of the memory block used in the Bayesian machines.** **a** Masks of the memory block with five routing metal layers. **b** Abstract view of the memory block, consisting of metal blockage masks to avoid routing above the memory block and pin masks to define the routing position of In and Out pins, and **c** the routing position of the supply voltages, which are easily distributed and routed.

conduct experimental verification on a fabricated chip to validate the functionality of our flow. As this was the first chip designed using our in-house automatic place and route design flow, successful experimental measurements can enable its use in future-generation designs.

We fabricated a fully-functional prototype circuit of a logarithmic Bayesian machine (see Fig. 3.6), using the hybrid CMOS/Memristor fabrication process (see Chapter 1), with 2,048 memristors and 35,400 transistors, using a the same special low-power 130-nanometer CMOS process as in Chapter 2, where hafnium-oxide memristors are fabricated in place of vias between metal layers 4 and 5.

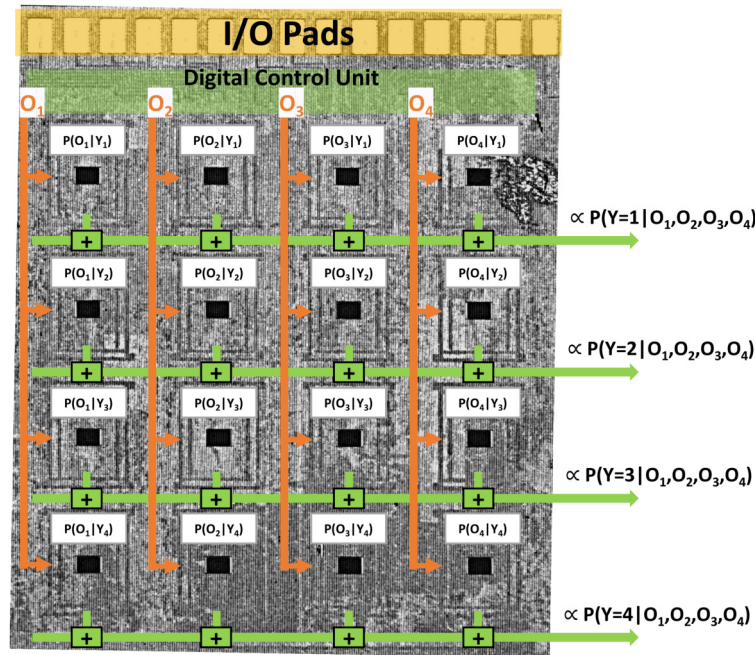


Figure 3.6: **Fabricated memristor-based logarithmic Bayesian machine.** Optical microscopy photograph of the Bayesian system die.

### 3.3 Measurements on the Logarithmic Bayesian machine

First, it is crucial to explore the forming and programming conditions of memristors, particularly given that the logarithmic Bayesian chip was fabricated during the next CMOS/memristor tapeout after the one of the stochastic Bayesian chip. With improved fabrication processes, the forming and programming voltages of new memristor-based chips may differ from those of previously tested chips, making it essential to investigate these conditions for optimal chip performance.

We designed an experimental setup at C2N, as illustrated in Fig.3.7, that incorporates a custom PCB to route an STM32 microcontroller unit with a probe station and power supply sources. The experimental setup was automated using Python scripts to streamline and expedite testing procedures. To conduct the measurements, we utilized an STM32 microcontroller unit to send and receive input and output signals from the non-packaged die, which was probed tested. This method is similar to the stochastic Bayesian machine described in Chapter 2.

After examining the forming and programming conditions, we obtained a likelihood memory array. We again verified that over 5 million consecutive readings with a 1.2 V supply voltage, no changes in memory values were detected. This finding led us to conclude that, similar to

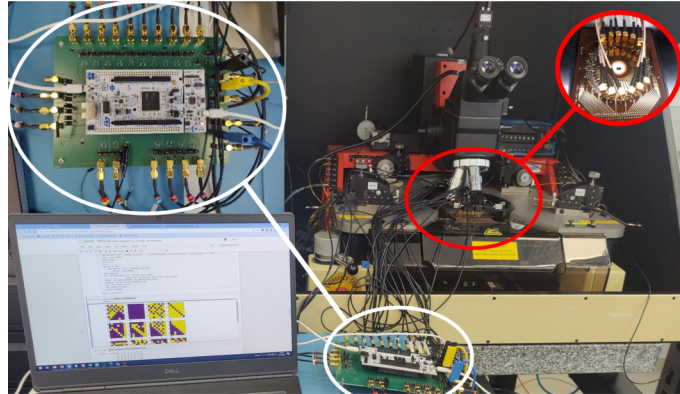


Figure 3.7: **The experimental setup for on-chip measurements on the logarithmic Bayesian machine.** The setup includes a custom PCB to route an MCU with a probe station and several power supply sources.

the stochastic case, the memristors did not suffer from read disturb issues.

In Fig. 3.8a,b, we present the inference measurement results for both the logarithmic and the stochastic Bayesian machines, respectively (the results for the stochastic machine are reproduced from Chapter 2, taken in the case with optimized LFSR seeds). The x-axis denotes the theoretical result anticipated from Bayes' law, while the y-axis denotes the experimentally measured results from the die's outputs. The different points in the Figure were obtained by randomly altering the observation inputs  $O_1$ ,  $O_2$ ,  $O_3$ , and  $O_4$  of the circuit (For the logarithmic circuit, some observations were chosen to better sweep all probabilities) and by varying the supply voltage  $V_{DD}$  between 0.65 and 1.2 V. Ideally, the experimental measurements should follow the  $x=y$  curve.

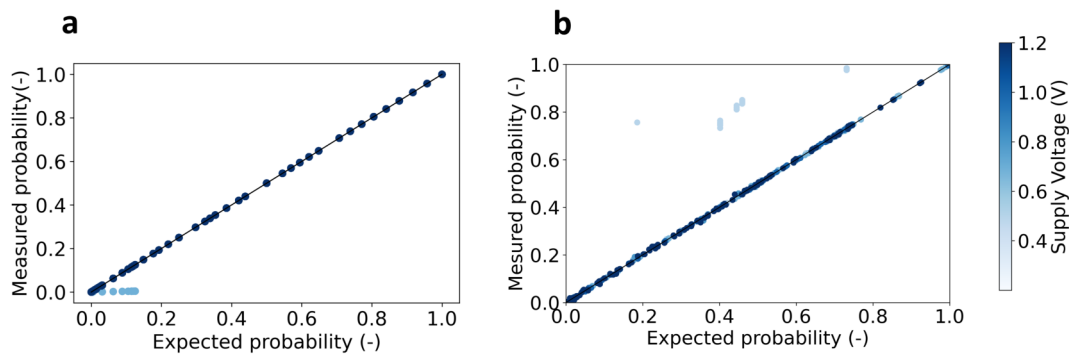


Figure 3.8: **Inference measurements on the fabricated Bayesian machines.** Measured output as a function of expected result on the fabricated **a** logarithmic and **b** stochastic Bayesian machine. In the logarithmic case, all points for supply voltages ranging from 0.7 to 1.2 V are superimposed.

As illustrated in Fig.3.8a,b, the measurements conducted on both chips display a remark-

able adherence to the ideal  $x=y$  curve for supply voltages ranging from 0.7 to 1.2V. The measured probabilities closely align with Bayes' law, with negligible deviations for measurements conducted with supply voltages ranging from 0.7 to 1.2V. Notably, in Fig. 3.8a, only the measurement points of supply voltage 1.2V are visible, as all measurements conducted within the range of 0.7 to 1.2 V yielded identical results, rendering them superimposed. (In the stochastic case of Fig. 3.8b, we chose to use different inputs for the different supply voltage to avoid this superimposition of results).

At supply voltages below 0.7V (light blue points), the measurements on both chips of Fig. 3.8a,b exhibit considerable deviations, resulting in less accurate Bayesian inference. This is attributed to the threshold voltage value of the thick oxide transistors used within the memory array, which is approximately 0.6 volts. Utilizing lower threshold-voltage transistors, available in many CMOS processes, could overcome this limitation. Interestingly, the errors manifest differently for each chip: while the stochastic Bayesian machine's errors can occur for any probability value, the logarithmic Bayesian machine only shows errors for small probabilities in the form of saturated values (represented as "1111111" due to overflow of the logarithmic representation). These errors cause the adders to detect overflow and saturate outputs. Ongoing investigations aim to uncover the reasons behind these errors and their occurrence specifically for small probabilities.

The accurate measurement results obtained from both Bayesian machines highlight their high potential. These machines are capable of producing precise outputs with high flexibility in terms of power supply. Furthermore, the systems remain fully functional even when reducing the power supply to 0.7 volts, which is a considerable reduction from the nominal supply of our CMOS technology, which is 1.2 volts. This feature allows for a significant reduction in power consumption, by a factor of approximately three.

The logarithmic machine provides an instant-on/instant-off feature, whereby the system is ready to perform Bayesian inference as soon as the power supply is turned on, without the need to load any data from memory. As a result, the power supply can be turned off anytime the system is not in use, without any penalty, offering an excellent opportunity for energy-saving.

Although both Bayesian machines share many design choices, they still differ in their computing approaches. Unlike the stochastic machine, the logarithmic machine does not require loading LFSR seeds or optimizing those seeds, making it easier to operate and immediately perform inference computation, thus reducing the complexity of the system's starting configuration. Additionally, the logarithmic machine performs inference in one clock cycle compared to the 255 clock cycles (i.e., the periodicity of the LFSRs) required for the stochastic machine, resulting in lower latency for inference. However, it takes eight clock cycles to read the output data from a logarithmic machine, compared to only one clock cycle needed for the stochastic machine. The readout latency for the logarithmic machine is due to the limited number of IO pads used in our chip; this issue is fixed in our next-generation chips.

Although the measurements show excellent results on both machines, the logarithmic ma-



chine produces more accurate results than the stochastic machine. This is because the stochastic computing is a conceptually approximate computing approach whose performance relies on the quality of the randomness generation, which is affected by used a pseudo-random number generator, leading to reduced quality of the randomness.

Regarding robustness, the stochastic machine is naturally resilient to soft errors. Bit errors can make one cycle wrong but will be averaged throughout the computation. This point is illustrated in Supplementary Note 9 of our published work [14]. However, the logarithmic machine is more prone to soft errors, especially when the errors occur in the most significant bits, which drastically change the probability values.

### 3.4 Energy Efficiency of the Bayesian Machines

The stochastic and logarithmic test chips utilized in our experiments serve as demonstrator chips that verify the feasibility of memristor-based Bayesian machines. However, their small size in terms of memory and computing capabilities prohibits their implementation in real-world tasks, and they are not suitable for accurate power consumption evaluations. In response, we developed a scaled-up version of the logarithmic Bayesian machine with  $6 \times 4$  likelihoods, each comprising 4,096 bits of memory, to perform a gesture recognition task, much like the approach we employed in Chapter 2 with the stochastic Bayesian machine. To accomplish this, we designed and laid out the system utilizing the reference low-power foundry 130-nanometer process and subsequently evaluated its energy consumption.

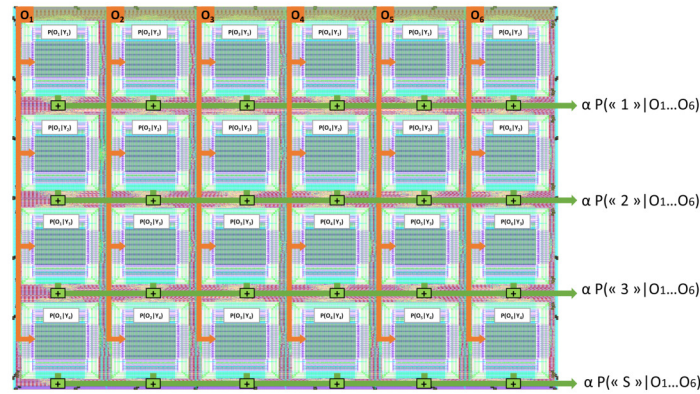


Figure 3.9: **Placed-and-Routed Logarithmic Bayesian Machine Masks for Gesture Recognition Analysis at the Design Level.**

Table 3.1 presents a comparison of the accuracy in gesture recognition and energy consumption for different situations for both stochastic and logarithmic Bayesian machines. The energy estimation method, based on the Cadence Voltus power integrity solution framework, is detailed in Chapter 2. This table utilizes the stochastic computing concepts introduced in Chapter 2. Conventional stochastic computing involves performing the computation for a certain number of cycles and then deciding based on the maximum number of ones. In the power-conscious mode, computation is stopped when the first one is encountered. Table 3.1 demonstrates that the logarithmic architecture performs better in accuracy, achieving 90.6% accuracy compared to 90% for conventional stochastic architecture and 86.9% for power-conscious stochastic architecture.

The logarithmic architecture also has a low energy consumption of 0.5 nJ, with the energy being dominated by the reading operation at 0.3 nJ (60%). Still, in terms of energy efficiency, the power-conscious stochastic architecture performs the best, consuming only 0.4 nJ with a full periodicity (255 clock cycles per inference), resulting in a 20% improvement in energy per-

formance over the logarithmic architecture at the cost of reducing accuracy to 86.9%. However, if the clock periodicity is reduced further, as in the case of 20 clock cycles per inference, the energy performance is improved to 0.34 nJ with a drastic loss in accuracy to 80.2%. This choice is not profitable, as it improves only the energy effect of the inference operation (0.04 nJ or 12% of total energy), which has limited impact on the total energy consumption of 0.34 nJ, which is dominated by read energy. On the other hand, the conventional stochastic architecture has accuracy comparable to the logarithmic architecture (90%) with a full periodicity (255 clock cycles per inference) but consumes considerably more energy, 2.47 nJ (5 times the energy of logarithmic inference). Reducing the clock periodicity to 25 clock cycles per inference results in energy performance similar to the logarithmic inference at 0.51 nJ with a drastic loss in accuracy to 82.9%.

Architecture	CLKs Inf.	Accuracy (%)	E (nJ) Inf.	E (nJ) Inf. & Rd
<b>Stoch Conv.</b>	255	90.0	2.17	2.47
<b>Stoch Conv.</b>	50	86.7	0.43	0.73
<b>Stoch Conv.</b>	25	82.9	0.21	0.51
<b>Stoch PC</b>	255	86.9	0.10	0.40
<b>Stoch PC</b>	50	84.4	0.06	0.36
<b>Stoch PC</b>	20	80.2	0.04	0.34
<b>Logarithmic</b>	1	90.6	0.20	0.50

Table 3.1: Comparison of the two Bayesian machines on the gesture recognition task. Conv: conventional stochastic computing. PC: power-conscious stochastic computing. Inf: inference

Based on the results presented in Table 3.1, we conclude that logarithmic computing performs better in energy consumption than both stochastic computing approaches, for accuracies higher than 86.9%. This is due to the fact that logarithmic computing reduces the inference operation to simple addition operations in one clock cycle, whereas stochastic inference requires multiple clk cycles with energy consumption dominated by random number generation, random number distribution, and clock distribution (as shown in the energy estimation results presented in Chapter 2).

We have shown in Chapter 2 that stochastic computing energy consumption could be reduced by using another type of random number generator. In our design, the consumption related to random number generation is 60% of the total consumption. However, even if we subtract this cost entirely (which is not a realistic assumption), we obtain an energy consumption of 1.18nJ ( $2.5 \times 0.4 + 0.3$ ). Logarithmic computing still outperforms conventional stochastic computing in terms of both accuracy and energy consumption for higher accuracies. In addition to its accuracy performance, logarithmic computing also has a shorter latency (one cycle) than stochastic computing. On the other hand, the stochastic machine has an inherent tolerance to single-event upsets (SEUs) [14].

### 3.5 Large Scale Multi-computing Mode Bayesian machine

Our Bayesian machine studies in this Chapter and in Chapter 2 have shown promising results based on measurements taken from demonstrator chips and energy estimates on scaled-up designs. These findings have inspired us to take the project further and fabricate a scaled-up system with larger memory and higher computing capacity that can handle real tasks on the chip. To achieve this goal, we implemented a large-scale chip that provides both logarithmic and stochastic computing modes. The logarithmic computing mode offers high accuracy at a lower computational cost, while the stochastic computing mode is well-suited for low-power, harsh environment applications. By using a multi-computing mode Bayesian machine, we can take advantage of the benefits of both computing modes in one chip. To make this possible, we used a mechanism that allows the Bayesian machine to switch between the logarithmic computing mode and stochastic computing mode depending on the specifications of the task.

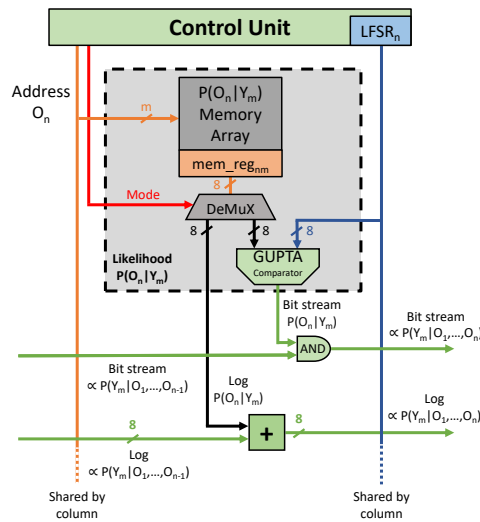


Figure 3.10: **Schematic illustrating the architecture of a multi-mode likelihood elementary block.** The likelihood elementary block used in the large-scale Bayesian version features two computing modes, the logarithmic and the stochastic. The mode needs to be decided before memristor programming.

In this section, we introduce our large-scale version of the Bayesian machine, which has been designed and taped out (see Fig. 3.11). This chip features 4x4 likelihoods elementary blocks with a 64x128 memristor array (8k-memristor) and approximately 143k memristor devices (131k for memory and 12k for dummies) and 285k transistors. This capacity is sufficient for testing real-life applications and enabling practical demos such as smart sensors, smart cameras, or simple robots.

Although both modes share the same memory arrays, read and write circuitry, the computing circuitry is separated. The stochastic version uses LFSRs circuitry and likelihoods with a comparator and AND gate circuitry, while the logarithmic version uses log-likelihood with the

adder circuitry (see Fig. 3.10). Since the probabilities are stored differently in each computing mode, the mode needs to be determined before memristor programming time.

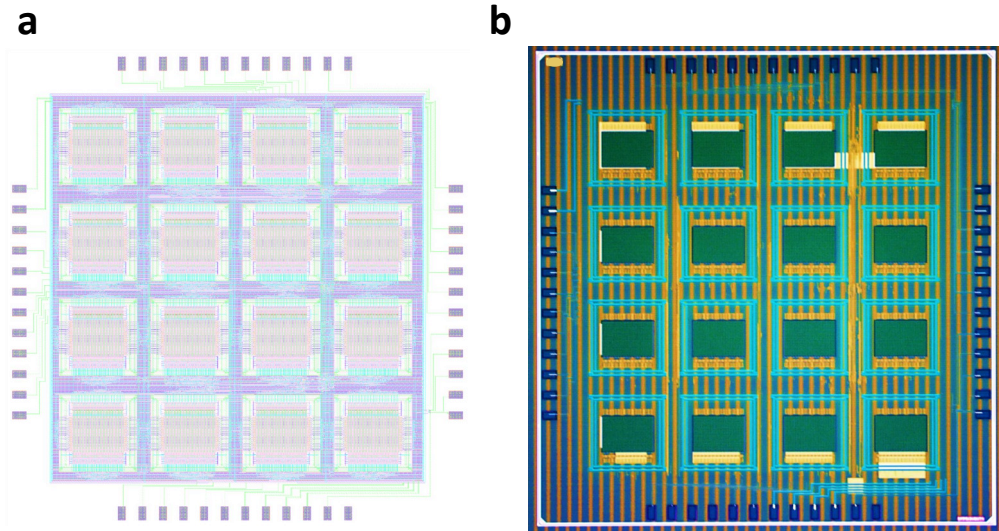


Figure 3.11: **The fabricated large-scale Bayesian machine.** **a** Masks of the placed-and-routed large-scale Bayesian machine design sent for fabrication. **b** Optical microscopy photograph of the large-scale Bayesian machine.

After successfully testing the packaged dies (with some dies damaged due to the packaging process), we were freed from the limitation of using only 25 IO pads in our designs (due to the custom probe card). As a result, the large-scale Bayesian machine includes 48 IO Pads, as shown in the layout view (see Fig.3.11a) and the optical-microscopy photograph (see Fig.3.11b). Therefore, the measurement setup for this chip is fixed to the packaged measurement option (see Chapter 1).

The circuit has already been fabricated, and at the time of writing of this thesis, it is currently undergoing the packaging process.

## 3.6 Conclusion

In this chapter, we have addressed the limitations of stochastic computing in memristor-based Bayesian machines and presented logarithmic computing as a solution to these limitations. We have shown that logarithmic computing offers increased precision and faster inference operations while retaining the same architecture and design choices as the general Bayesian machine (see Table 3.2). We have demonstrated that the logarithmic Bayesian machine can be implemented with distributed memristors, performing computation locally and with minimal energy movement. The logarithmic machine also inherits all the advantages of the Bayesian inference approach, such as explainable models, uncertainty information, and training with limited data. Our new approach targets being embeddable at the edge, with low power consumption, enabling edge systems to benefit from the qualities of Bayesian inference to deal with highly uncertain situations with little data and to make predictions using an explainable mode.

We reported the design and fabrication of the logarithmic Bayesian inference circuits, then the testing of the two prototype Bayesian inference circuits – the stochastic and logarithmic computing ones. We showed that both machines have accurate measurement results, highlighting the high potential of our Bayesian inference machines, with high flexibility in terms of supply voltage. Using a homemade energy estimation framework, we showed that both designs can perform a gesture recognition task using orders of magnitude less energy than a microcontroller unit. Each design can be more suitable for specific applications based on energy and accuracy constraints. Based on the energy estimation results, we conclude that logarithmic computing performs better in energy consumption than both stochastic computing approaches, for accuracy higher than 86.9%. Stochastic computing is more energy-efficient for lower-accuracy inference up to 86.9% for our gesture recognition task, due to the power-conscious inference strategy.

Our results show that memristor-based near-memory Bayesian computing is a viable solution for energy-efficient machine learning systems. These results highlight the potential of memristor-based near-memory Bayesian computing, even with inexpensive technology such as the 130-nanometer process we utilized. Scaling up the design to more advanced technology nodes can help to further reduce energy consumption.

Overall, the logarithmic Bayesian machine approach provides a promising solution to the limitations of stochastic computing in memristor-based Bayesian machines. The potential of this approach for energy-efficient and accurate machine learning systems motivates us to create a larger and more powerful system capable of implementing real tasks on the chip.

<b>Features</b>	<b>Stochastic Machine</b>	<b>Logarithmic Machine</b>
Area Requirements	Minimal	Minimal
Energy Consumption	Low	Low
Precision	Limited	Increased
Inference Speed	Slow	Fast
Complexity with RNG	Yes	No
Suitability for Bayesian Inf.	Yes	Better
Underflow and Loss of Precision	Yes	Less
Soft Error Resilience	Resilient	Susceptible
Ease of Inf. Operation	LFSRs Configuration	No Configuration
Inference Time	255 clock cycles	1 clock cycle
Accuracy	Near-Perfect	Accurate
Energy for Accuracy > 86.9%	More	Less
Energy for Accuracy < 86.9%	Less	More

Table 3.2: Comparison between Stochastic and Logarithmic Machines

## Chapter 4

# Multimode Memristor-based Prototyping Platform

If it disagrees with experiment, it's wrong. In that simple statement is the key to science.

---

Richard FEYNMAN



Memristors offer fantastic opportunities in the field of computing for implementing new paradigms [243, 244], such as analog computing[245], neuromorphic computing[104], stochastic computing [228], and In or Near memory computing [11, 246]. However, memristors are based on emerging technologies that are still in the research and exploration stage, and several challenges and imperfections need to be addressed. Such challenges emphasize the importance of experimental platforms for prototyping new computing paradigms. These platforms enable the implementation, exploration, and experimental validation of new ideas, as well as the validation and optimization of reading, programming, and computing techniques.

The purpose of this chapter is to present an integrated circuit that provides a prototyping platform for projects involving memristors. This circuit includes periphery circuitry for using memristors within digital circuits and an analog mode with direct access to memristors. This platform allows for developing and testing innovative memristor-based neuromorphic concepts that address specific challenges and requirements.

In the following sections, we will first discuss memristor imperfections and the challenges in implementing new computing paradigms using memristors. This discussion will naturally lead to the potential solutions for using imperfect memristors and the importance of memristor-based experimental platforms. Then we discuss the design, fabrication, testing, and potential projects for our experimental multimode memristor-based platform.

The integrated circuit presented in this chapter was presented at The ASPDAC 2023 conference [87].

## 4.1 Imperfect Memristors for Building New Computing Paradigms

Data storage in physical devices relies on underlying physical phenomena, such as magnetic or electronic processes. To ensure accurate programming, retention, modification, and readout of device states, these phenomena must be well understood and controllable. Different memory technologies rely on distinct physical phenomena and use varied approaches and methods. For example, in volatile memories like DRAM and SRAM, data is stored by manipulating electric charge movement and storage. On the other hand, mature non-volatile memories such as flash memory utilize the charge storage method, wherein the floating gate is charged with trapped electrons using the quantum tunneling phenomenon to shift the transistor threshold voltage and implement the memory effect.

Emerging non-volatile memory technologies like memristors control the resistance state of the device to store data. The resistive switching mechanism differs depending on the specific technology used. For example, OxRAM-based RRAM or memristor uses oxygen vacancy filaments in the oxide layer to store data, while PCM uses changes in the material phase to store data. MTJ and FRAM use spin orientation of magnetic fields and electric polarization of ferroelectric materials, respectively, to store and retrieve data.

However, due to the emerging nature of memristor technologies, challenges and imperfections still exist. For example, OxRAM-based memristors exhibit non-linearity, asymmetry, instability (drift), and variability. These issues can negatively impact performance, increase energy consumption, and make system design more complex. In this section, we will discuss the non-idealities and imperfections of OxRAM-based memristors, the implications of these imperfections for non-conventional computing, and potential solutions to mitigate these issues. Finally, we will explore new computing models that embrace imperfection and leverage non-idealities.

### 4.1.1 Non-Ideal Behavior of OxRAM-Based Memristor

The main operation for programming OxRAM devices involves the SET and RESET processes, which result in resistance switching through the formation and dissolution of the oxygen vacancy-based conducting filament (see Fig. 4.1a). However, the exact mechanism underlying the switching process is not fully understood. This has led to several challenges in exploiting these devices for both storage and computing applications.

One such challenge is the asymmetry in the SET and RESET programming processes, as shown in Fig. 4.1a, where the current-voltage (I-V) characteristics of the memristor is dependent on the direction of the applied voltage. This asymmetry results in up and down conductance changes that are not directly symmetrical, making it difficult to direct the switching between states. Especially for multi-level states, programming memristor typically requires extra

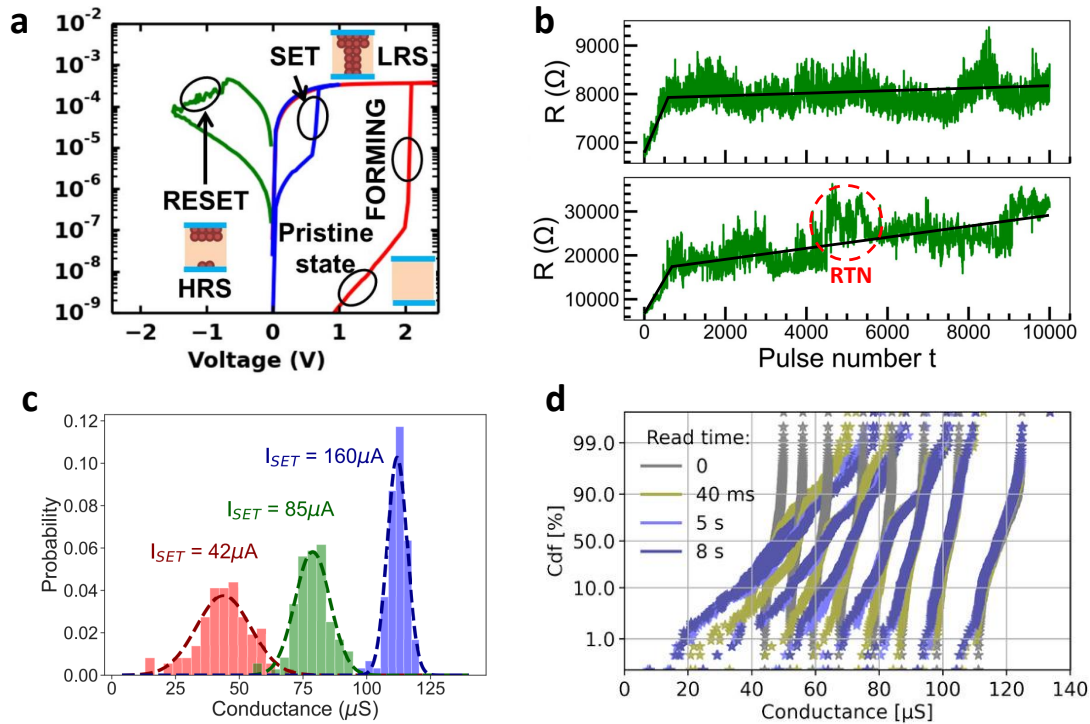


Figure 4.1: **Imperfections of OxRAM-based memristors.** **a** Current–voltage OxRAM characteristics for FORMING, SET, and RESET operations. An asymmetry in the SET and RESET programming voltages is seen (reproduced from [22]). **b** Progressive evolution of the resistance of two measured devices with consecutive weak RESET pulses. We see non-linearity and instability of the resistance change with consecutive applied voltage (reproduced from [23]). **c** Cycle-to-cycle programming variability in resistance states, Distribution of the low resistance state for different SET programming conditions (reproduced from [24]). **d** Cumulative distributions of OxRAM devices in eight different conductance levels, after standard iterative programming, a resistance drift can be seen (reproduced from [25]).

set and reset operations to achieve the desired state.

Another challenge is non-linearity, which refers to the fact that the resistance of an OxRAM-based memristor does not change in a linear fashion with applied voltage, as shown in Fig. 4.1b. The resistive states exhibit stochastic and non-linear incremental resistance changes, with two progressive increase regimes observed in the cell resistance: an initial more progressive increase followed by a subsequent noisier increase that suffers from Random Telegraph Noise (RTN). This result confirms the idea that higher resistance states are less stable. During the reading operation, a noisy reading of the resistance state may occur when a steady reading current is applied over a period of time, which can also disturb the device state if the reading conditions are harsh.

The OxRAM is advantageous because it can be programmed in single-level or multiple-level states in a non-volatile fashion. However, resistive states obtained after programming are im-

perfect even under identical programming conditions. This imperfection is called variability, with two main types distinguished in the OxRAM: device-to-device variability, which refers to the variation between different devices, as seen in Fig. 4.1b, where the progress of the resistance state of two devices differs even under the same programming conditions; and cycle-to-cycle programming variability, which refers to the programmed resistance state differing over successive programming cycles for the same device. Fig. 4.1c shows the probability distributions of cycle-to-cycle conductance variability of a single device programmed under three different SET programming conditions, with the distributions fitted using a normal distribution.

Even if the programming process is optimized to achieve ideal resistive states, OxRAM-based memristors are still prone to resistance instability, called the drift effect, which refers to the fact that the resistance state of the memristor can change over time, even without any external stimuli, potentially leading to data loss. This effect can be more pronounced in the high resistance state (or low conductance state, LCS), as shown in Fig. 4.1d.

Despite the non-idealities and imperfections present in memristors, they are still considered a highly promising technology for energy-efficient computing systems. Ongoing research and development aim to better understand the switching mechanism, mitigate the impact of imperfections, and even leverage them for improved performance. Various techniques for characterizing and modeling imperfect memristors have been developed, and novel algorithms, architectures, and techniques are being explored. Understanding and mitigating the impact of imperfections in memristors is crucial for building new computing paradigms.

### 4.1.2 Mitigating Imperfections for Non-Conventional Computing

Imperfections and non-ideal behaviors of OxRAMs have implications for using those devices on both conventional and non-conventional computing paradigms. If those devices are to be utilized as a conventional digital storage memory, the programming imperfection, both Single-Level Cell or Multi-Level Cell, can be solved using conventional EECs. However, EECs are not compatible solutions for near-memory computing. For this type of computing, other solutions have been suggested, such as using 2T2R bit-cell structure with complimentary programming as we did in our Bayesian machines (Chapters 2 and 3). Another solution is using 1T1R structure with intense programming conditions to increase the gap between LRS and HRS for Single-Level Cell storage, with the cost of decreasing device endurance, or the use of iterative programming strategy to improve the programmed states distribution.

We have also seen in Chapter 1 that memristors are being explored for in-memory computing, which involves performing computational tasks directly within the memory devices, rather than in a separate processing unit. In particular, memristors can implement multiply and accumulate (MAC) operations (deep learning's basic operation) in an analog fashion, relying only on Kirchhoff laws. However, variability has a high impact on this type of computing, as it leads to decreasing the computing accuracy. This is because memristor's conductance state is used for both analog storage and analog computing: this limits the possibility of using error correcting

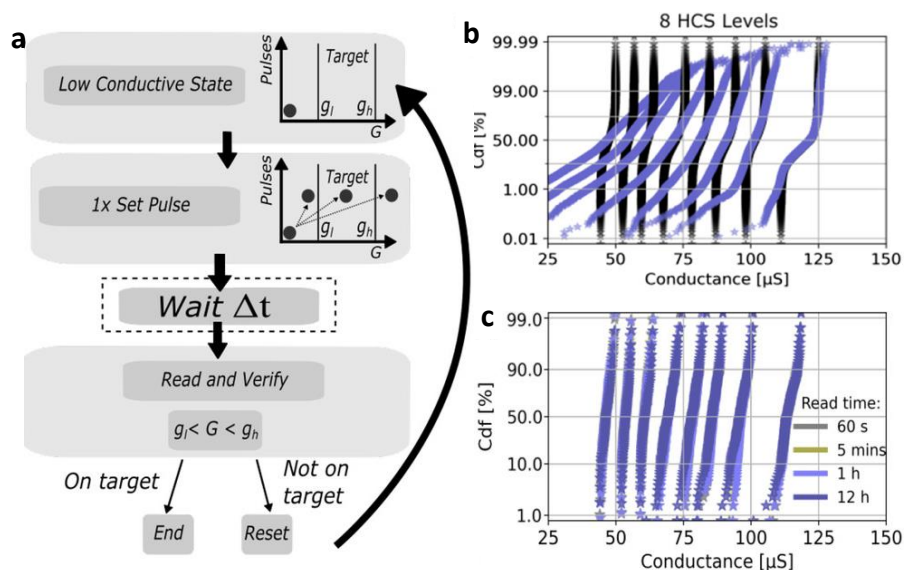


Figure 4.2: **Optimizing memristor programming algorithms.** **a** Optimized iterative programming algorithm. **b** Conductance cumulative probability distribution for eight distinct conductance levels programmed using the standard iterative programming algorithm. **c** Conductance cumulative probability distributions for eight conductance levels programmed using the optimized programming technique in **a**, stable resistance states read after 60s and 12h. (reproduced from [25])

techniques. Therefore, the need for more accurate and stable analog states is an important aspect. To overcome the programming variability problems, an iterative programming algorithm was used for the purpose of the programmed conductance state distribution (decreasing the distribution's STD). This algorithm relies on a program-and-verify strategy: memristor devices are programmed multiple times, with a target conductance interval. This programming strategy has successfully improved the multi-state device programming. In Fig. 4.2b (black curve), we can see the separated programmed conductance states. However, this is only a short-term strategy, as OxRAM conductance states suffer from instability over time 'drift effect', as it can be seen Fig. 4.2b (blue curve) reading the devices after 60 s from programming, the distributions worsen, specially for the high resistance states.

To overcome the conductance instability effect (conductance drift), an optimized program-and-verify technique was recently proposed (see Fig. 4.2a), with the addition of a wait time of  $\delta t$  before the conductance verification step. This technique ensures that the verification step will check both cases: the imperfect programmed devices and the unstable drifted states before the next program iteration. Fig. 4.2c shows that the resulting programmed cells have highly stable states: the conductance distributions were stable after 1 min and after 12 h. Overall, the non-ideal behaviors such as variability of memristors can potentially be compensated with optimised programming techniques or a proper design choices of the bit-cells. However, those optimizations need to be adopted in hardware, and this requires changes in the periph-

eral circuitry, which will lead, typically, in increased circuit complexity and operational time and energy. All of those compromises need to be studied to confirm the advantages of the non-conventional computing in terms of energy and performance than the equivalent digital implementations.

Finally, in neuromorphic computing, the field of research that seeks to develop brain-inspired artificial intelligence systems, memristors have been proposed as a key element due to their ability to implement artificial synapses. However, the non-ideal characteristics of memristors can pose challenges to implementing accurate and reliable neural networks, especially to perform learning on chip. As synaptic or weight values are stored in a form of a conductance or resistance, the non-linearity and the asymmetry effects make changing device resistance or conductance during learning very difficult, as resistance variation is not the same when applying positive and negative voltage pulses. These imperfections are problematic as learning rules tend to require uniform and precise weight update operations (Fig. 4.3c). The presence of noise or device-to-device variability is contradictory to that.

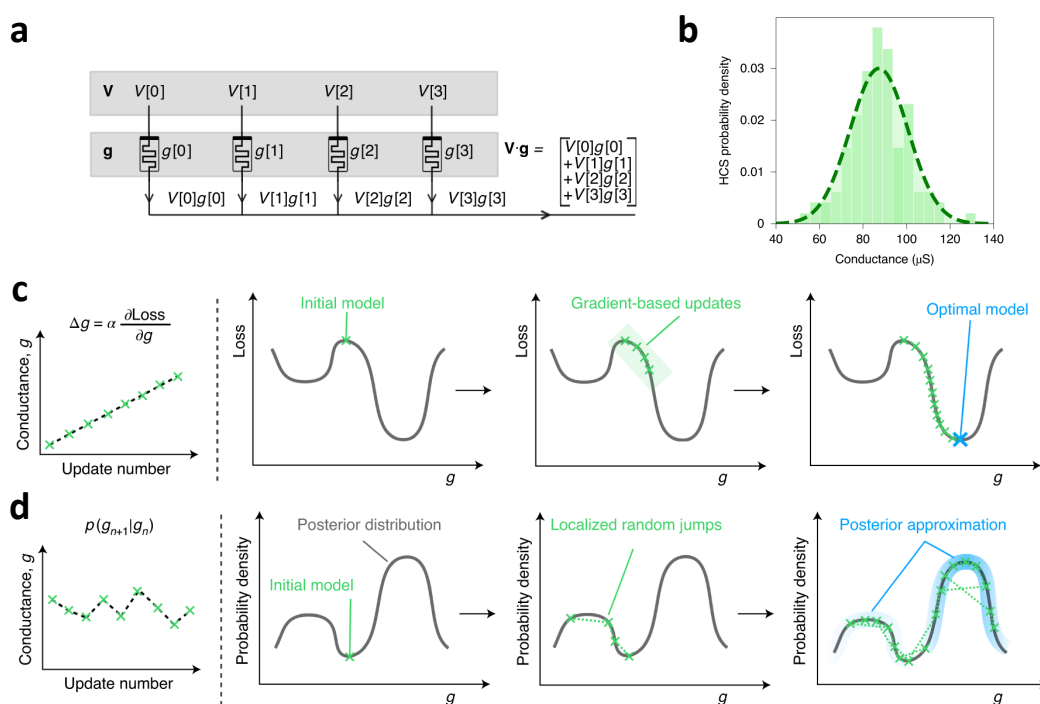
One approach to mitigating the impact of imperfections in memristors for neuromorphic computing is to develop novel algorithms and architectures that are inherently robust to imperfections. A hardware-aware algorithm is needed, requiring a co-design approach between hardware and software. Overall, while imperfections in memristors can pose challenges for building reliable computing systems, researchers are actively working on developing novel approaches to mitigate these imperfections and unlock the full potential of memristors for non-conventional computing applications.

### 4.1.3 Embracing Imperfection for Non-Conventional Computing

Rather than trying to eliminate the imperfections of memristors, some research embraces those imperfections and explores new computing approaches that leverage non-idealities. These approaches aim to take advantage of the inherent variability, non-linearity, and complexity of memristors to enable new computing paradigms that are more efficient, robust, and adaptable. Some examples of such models include neural networks with stochastic synapses, brain-inspired computing systems, and reservoir computing.

In this subsection, we report an example of embracing the device imperfection [88]. We present a project where we were collaborating with a team from CEA-leti, about a novel machine learning approach that exploits the variability of memristors to implement in-memory Markov chain Monte Carlo (MCMC) sampling algorithms [247] in a fabricated array of 16,384 devices configured as a Bayesian machine learning model. The algorithmic and experimental work was performed mainly by first author Thomas Dalgaty, while I contributed to system-level evaluations and benchmarking aspects of the paper. The approach is experimentally demonstrated for tasks such as malignant tissue recognition, heart arrhythmia detection, and the cartpole reinforcement learning task. We showed that cycle-to-cycle conductance variability in memristors (see Fig. 4.3b) can be viewed as a physical random variables, which offers ran-

dom conductance updates deriving from a known probability distribution. This feature allows memristors to perform in-situ MCMC sampling operations; the random samples make localized random jumps on the posterior distribution (see Fig. 4.3d), giving us an approximation of the posterior distribution. The Bayesian model parameters are stored in the same devices as conductance states. This eliminates the need to transport information between processing and memory. Those Bayesian conductance-based models,  $\mathbf{g}$ , can be inferred by performing the analog dot product between the input voltage vector  $\mathbf{V}$ , and the conductance vector  $\mathbf{g}$  resulting in a current flow equivalent to  $\mathbf{V}\cdot\mathbf{g}$  (see Fig 4.3 a).



**Figure 4.3: Analog in-memory computing with imperfect memristors.** **a** A single array row that can perform in analog fashion a dot product  $V\cdot g$  operation. **b** Probability density of the cycle-to-cycle variability for a single memristor. It follows a normal distribution, which makes memristors serve as a random variable. **c** (left) Gradient-based learning algorithms iteratively compute the derivative of an error metric with respect to a conductance model  $g$ , multiplied by a learning rate  $\alpha$ , to determine updates to be applied to the  $g$  parameters. The ideal memristor device should be capable of high precision and linear conductance updates. (right) The three panels show the gradient descent algorithm for an increasing number of model updates (green crosses). From an initial model, the algorithm performs gradient-based updates until it converges to a local minimum in error. **d** In our work, memristor random conductance updates are used by the sampling algorithm to perform, local random jumps on the posterior distribution, then an approximation of this distribution is stored.

To prove the concept of memristor-based MCMC sampling, we implement an experimen-

tal system that consists of a computer-in-the-loop with a fabricated memristor array with 1T1R configuration. We train the system to recognize cancerous mammographies. Benchmarked against deterministic software-based neural network models, we find that the resulting Bayesian models perform better, and that memristor-based MCMC trains a full model with orders of magnitude fewer programming operations compared to existing memristor-based backpropagation approaches. Finally, through the design and simulation of a fully-integrated implementation of our approach, we compare the training energy of our approach with that required using only CMOS technology with conventional architecture approaches (MCU), and observe an energy reduction of several orders of magnitude. Our approach could also support the implementation of several Bayesian learning algorithms. This prospect is supported by the fact that Bayesian network topologies are already employed in some biomedical machine learning applications [248].

After the publication of our work [88], I continued to work on the design of a standalone and fully integrated memristor-based MCMC sampling chip, which incorporates additionally to the memristor array, analog programming, and inference circuitry. Using a new 2T1R memory configuration (see Fig 4.4c), the chip can perform the sampling operation by realizing sensing and programming simultaneously. The chip has been designed, fabricated, and it is at the packaging stage now (this chip is not directly related to the one described later in this chapter).

A common feature of all ongoing research and development for the purpose of exploring, optimizing, mitigating the impact of imperfections, or leveraging them for improved performance is the need for an experimental platform. Our research and projects on those subjects were due to a successful collaboration with teams that own experimental platforms. The increased need for experimental validation of our ideas and for accelerating the research process pushed us to work on developing an internal experimental platform with the help of the expertise of our collaborators.



## 4.2 Description of the Hybrid CMOS/Memristor Die

Memristor-based computing projects are made easier through the use of simulations based on memristor behavioral models. Alas, these models fail to provide a perfect description of memristor non-ideal behaviors and imperfections for all programming conditions. As each project involves the exploration of new ideas, specific programming conditions and strategies are required. This means that researchers can either use a specific model for each project or rely on accessible experimental data based on desired programming conditions from a collaborator. Alternatively, they can employ an experimental platform for the on-chip implementation of new ideas. While the latter is the most challenging option, the use of memristor-based experimental platforms is an essential tool for researchers exploring and validating new computing paradigms. Experiments allow us to collect high-quality data that would be difficult or impossible to obtain otherwise, providing precise measurements and insights into the underlying mechanisms that govern complex phenomena. Moreover, experimental platforms enable to optimize the conditions and techniques for reading, programming, and computing using memristors, as well as developing and testing innovative memristor-based neuromorphic concepts that address specific challenges and requirements. Although it can be challenging to design, fabricate, or gain access to cutting-edge technologies, the benefits of utilizing experimental platforms cannot be overstated.

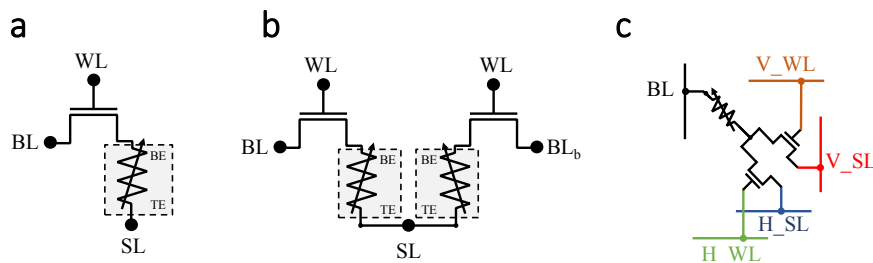


Figure 4.4: **Memory cell structures.** **a** 1 Transistor 1 Resistor (1T1R) structure, for analog mode computing. **b** 2 Transistor 2 Resistor (1T1R) structure, for digital mode computing. **c** 1 Transistor 1 Resistor (1T1R) structure, for MCMC sampling.

The design choices for memristor-based integrated circuits (ICs) used in research depend on the specific goals of the project. For instance, the memristor-based IC can be designed as a memristor array serving as a platform for characterization, analog computing, or implementation of novel ideas such as in-situ learning algorithms. This approach offers researchers increased flexibility, allowing direct access to memristor devices while utilizing the computational power of a computer for implementing reconfigurable logic and arithmetic functions. Alternatively, for projects focused on demonstrating performance, complete memristor-based Application-Specific Integrated Circuits (ASICs) can be developed to test and analyze the performance of memristor-based ICs for specific tasks, such as artificial neural networks [249] and



designed using thin oxide low-power transistors and supplied by digital nominal voltage (except for level shifters). Orange-colored blocks are analog-mode circuits, also designed using thick oxide transistors.

### 4.2.1 Digital Mode Circuitry

The digital mode circuitry (Fig 4.6a) mirrors the read and program strategies used for the Bayesian machines, as detailed in Chapters 2 and 3. In addition to the digital read and program circuits—which include row and column decoders, level shifters, and precharge sense amplifiers—each column incorporates a logic-in-memory feature based on an idea originally introduced in [222].

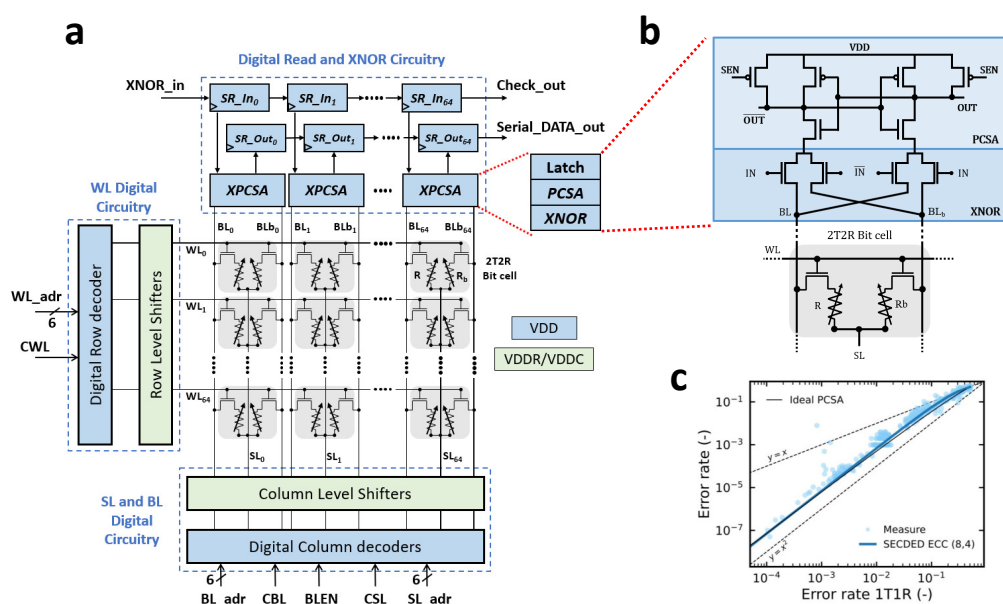


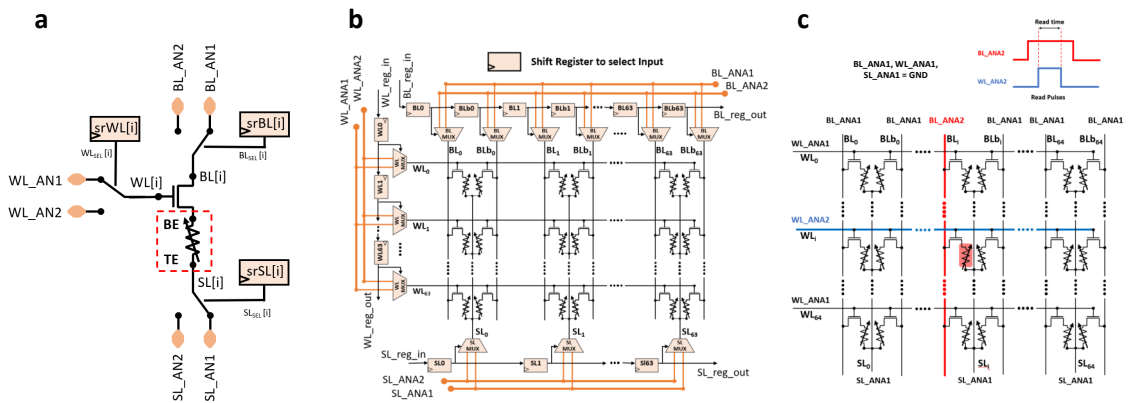
Figure 4.6: **Digital Mode Circuitry.** **a** Schematic of the digital mode circuitry. It consists of address decoders, level shifters, PCSA sense and XNOR compute circuitry, input and output shift registers. The design adopts the 2T2R structure for storing one bit. **b** Schematics of the sensing circuitry with XNOR logic-in-memory feature, the pass transistor logic XNOR, the differential precharge sense amplifier used to read the binary memristor states and the SR Latch. **c** Error rate of the 2T2R approach as a function of the error rate of the 1T1R approach, in simulations assuming a perfect PCSA (black line) or experimentally measured on the integrated circuit of [26] (light blue points). Blue line: error rate of a SECCDED ECC using the same number of devices as our 2T2R

The precharge sense amplifiers allow reading the binary states of memory cells in a highly energy-efficient fashion while optionally performing XNOR logic operations at the same time (Fig 4.6 b). The 2T2R memory cell with complementary approach of [220] is used in our array for reducing the bit error rate. The total digital memory that can be stored with the complemen-

tary approach is 4 kilobits. Because of the limitation in the number of IO pads, shift registers are used for both logic-in-memory inputs and Data Outputs, creating limitations for data read latency.

The efficiency of the 2T2R approach to reduce bit errors is confirmed in Fig 4.6c. This Figure plots the bit error rate of the 2T2R approach as a function of the one of the 1T1R approach, obtained experimentally and theoretically. The theoretical result assumes a perfect sense amplifier. The experimental results are reproduced from [26] and were obtained on a different integrated circuit based on the same memristor technology. We also plotted the error rate that would be obtained when using a conventional Single Error Correcting/Double Error Detecting correction code (SECDED, or extended Hamming) that doubles the number of memristors, as our approach. We see that our approach reduces the number of bit errors almost equivalently to the SECDED code. However, the SECDED code requires area, delay, and energy-costly error decoding circuits, while, in our approach, error correction happens naturally within the sense amplifier without any additional cost.

### 4.2.2 Analog Mode Circuitry



**Figure 4.7: Analog Mode Circuitry.** **a** Simplified schematic of 1T1R cell in analog mode, illustrating the switching of analog InOuts. **b** Schematic of the analog mode circuitry, with shift registers selecting inputs via Multiplexers, which consist of analog MUXs connected to SL, BL, and WL terminals. Each MUX is controlled by a shift register, to choose one of two analog inputs. **c** An example of measurement of memristor conductance from the memristor array.

When activating the analog mode, digital circuits are deactivated, and the memristors array connections are switched to the analog circuitry (Fig. 4.7a). In this mode, shift registers configure input multiplexers permitting direct access to the analog state of memristors, using low-resistance transmission gates used in the analog MUXs (Fig 4.7b). Hence, shift registers have an equivalent function of addressing the accessed device. Each word line, bit line, and source line is then connected to one of two analog InOut Pads (most of time, one of two ana-

log InOuts is connected to the ground), which can be connected to external equipment, e.g., Keysight B1530, a pulse source and measurement unit widely used to characterize memory devices. Fig. 4.7c shows an example of configuration allowing the measurement of a memristor conductance from the array.

The analog mode has access to the 8k memristor devices that can serve as analog storage. However, because we are limited to only two analog inputs, the array can implement analog MAC operation only for Binarized Neural Network. Multi-level analog MAC operation can be implemented virtually using a computer-in-the-loop experiment, based on the measurements on each device.

### 4.2.3 Design Signoff and Measurement Setup

The memristor array and all analog and mixed-signal circuits were designed in a full custom fashion. All digital circuits were placed and routed automatically using an HDL description and the Cadence Encounter flow. Then, all circuits of the system were assembled manually and routed automatically using a Cadence Encounter flow developed in-house using a homemade abstract view of the memory array (see Chapter 1).

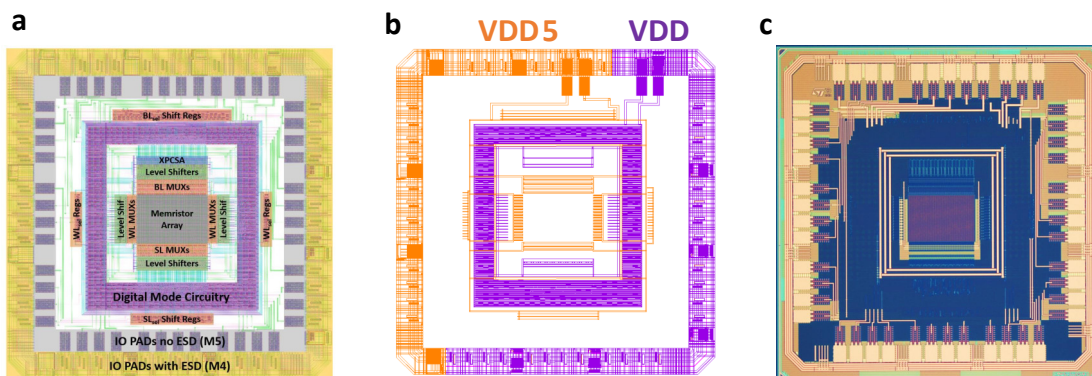


Figure 4.8: **Fabricated Multimode Hybrid Memristor-CMOS Prototyping Platform.** **a** layout view, **b** Supply voltages connections and **c** Optical microscopy photograph.

A photograph and a layout view of our integrated circuit are presented in Figs. 4.8a-c. To reduce the risk of damaging the dies during the packaging, we integrated complete ESD protection in this chip. We used two sets of IO pads: foundry pads, which embed appropriate ESD protection (Figs. 4.8a, yellow region) but that are limited to metal four, and the IO pads provided by our partner with metal five layer (Figs. 4.8a, grey region). This is because the die needs to be post-processed by our CEA-Leti partner to add memristors and metal five; then it is wirebonded at metal level five during the packaging process.

Another important design choice for this chip was to have a full separation between the supply voltages of digital circuitry and the supply voltages of the analog circuitry, from device to the IO pads level (Figs. 4.8b). The advantage here is to reduce any kind of interference between

the modes during measurements.

To make the system re-configurable for different projects, we developed the experimental setup of Fig. 4.9: a PCB routes a microcontroller unit and measurement equipment with our packaged die. To make the platform accessible to users without extensive knowledge of electronics, digital and mixed-signal circuits, interfacing Python scripts were developed to implement the basic analog and digital operations, such as read and program devices from the array. Those scripts control all the measurement equipment, and the user only needs to develop a specific script for implementing the functions related to the defined experiment.

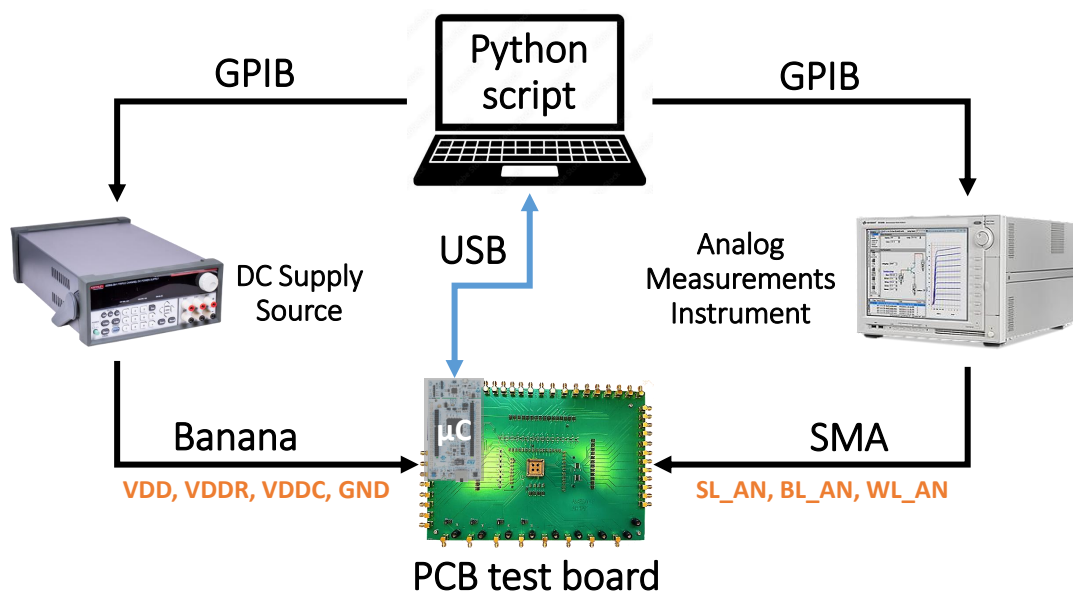


Figure 4.9: Measurements setup of the prototyping platform.

## 4.3 Uses of the Platform

This section briefly discusses the potential applications of our prototyping platform for both analog and digital computing projects, highlighting its versatility and potential impact for future research.

### 4.3.1 Digital Prototyping Projects

Several potential digital computing projects could be implemented on our platform. Optimizing read and programming strategies using the digital mode can allow the successful implementation of digital applications. Memristors feature a complex interplay between programming energy, reading speed, read disturb effects, and device endurance, which our platform allows understanding.

A Binarized Neural Network (BNN) can be implemented using memristors as digital storage and the in-memory XNOR circuitry from our platform to implement the BNN multiplication [250]; with adding memory digital or analog popcount [251] and threshold circuitry outside of the platform, all needed computation are fulfilled.

Synaptic metaplasticity in binarized neural networks [252] is another interesting project that could be implemented using the platform, for attempting to overcome the “catastrophic forgetting” problem of neural networks. The hidden weights used by binarized neural networks, can be used as metaplastic variables, and memristors can store the analog values of those weights.

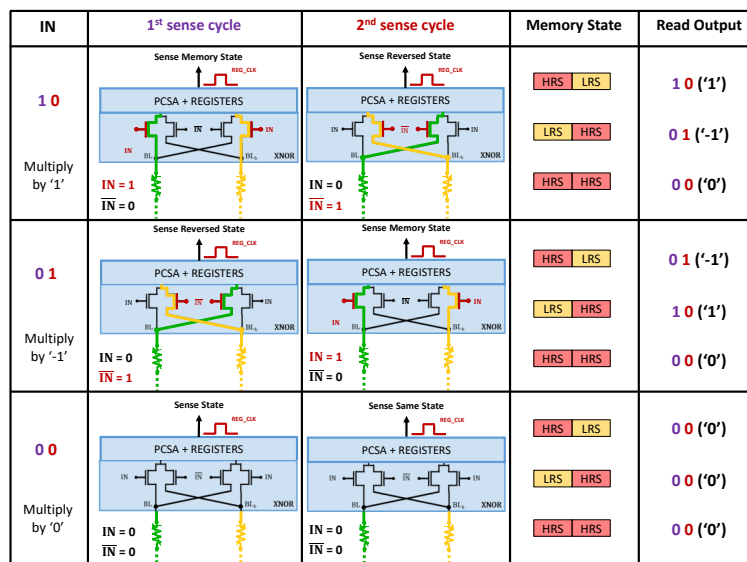


Figure 4.10: **Principle of logic-in-memory ternary input weight multiplication.** The input is voltage  $IN$ , the weight is stored in the 2T2R cells, the multiplication is done in two cycles.

Another project idea concerns memristor-based Ternary Neural Networks (TNN). It would extend the basic approach of storing ternary values in the 2T2R structure shown in [253]. In this approach, weights are programmed using two memristors per weight. Discharge rates reveal memristor states: slow discharge occurs when both memristors are in high resistance state (HRS), representing zero weight; if one memristor is in low resistance state (LRS) and the other in HRS, either the output or the complimentary discharges in a few ns depending on the programmed weight being 1 or -1. I have proposed an update of this concept, which can be implemented in the platform, where this behavior enables logic-in-memory operations in two cycles (see table in Fig. 4.10): the sense amplifier output reflects the product of input IN and the programmed weight.

### 4.3.2 Analog Prototyping Projects

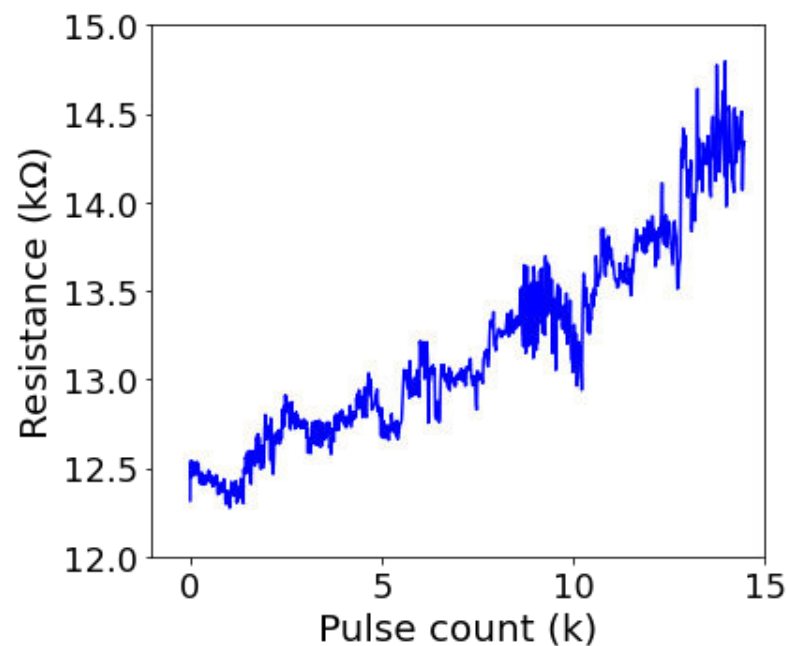


Figure 4.11: **Measurement of memristor resistance as a function of number of RESET programming pulses.** A characterization experiment for implementing a synaptic learning rule.

The analog mode of the platform can be used to prototype computing concepts where memristors are used in an analog fashion, e.g., as artificial synapses in machine learning or neuromorphic circuits [103]. Fig. 4.11 shows measurements on a memristor in our platform when applying a succession of 15,000 1V 1.5- $\mu$ s programming pulses: the memristor resistance progressively increases, a feature that permits the memristor to implement a synaptic learning rule. This use is particularly appealing due to its compactness, but the imperfections of memristors (thermal and random telegraph noise, cycle-to-cycle, and device-to-device variability)



pose challenges that make it necessary to test ideas experimentally. Our platform supports prototyping various neuromorphic experiments, targeting inference, deterministic or probabilistic learning [88].

In fact, a Ph.D. student of the group, Marie Drouhin, is currently utilizing the platform to demonstrate the training of a memristor-based neural network using equilibrium propagation, a hardware-friendly alternative to the widely-used backpropagation algorithm [254]. Equilibrium propagation addresses some of the hardware limitations associated with backpropagation, such as the need for complex weight update computations and excessive data movement, making it an attractive choice for implementation on neuromorphic hardware like our platform. The student's research aims to showcase the potential of memristor-based neuromorphic systems in efficiently executing learning tasks and to provide insights into the practical challenges and performance trade-offs associated with this innovative approach.

The concept of mortal computation and the new Forward-Forward Algorithm (FFA) have been recently suggested by Geoffrey Hinton [255]. These new concepts have promising implications for the future of computing, as they offer hardware-friendly algorithms, energy, and cost savings. Memristors are particularly well-suited for implementing the FFA and supporting mortal computation. Our memristor-based experimental platform is an ideal candidate for projects of implementing the FFA, as it allows for the exploration and optimization of the unique properties of individual memristors.

## 4.4 Conclusion

In conclusion, memristors offer unique characteristics that make them a promising hardware solution for implementing new computing paradigms. However, several challenges, memristor imperfections, need to be addressed to fully utilize their potential. Experimental platforms are crucial for addressing these challenges and developing and testing innovative memristor-based concepts.

We have presented a multimode memristor-based prototyping platform that enables the implementation and exploration of new ideas and the validation and optimization of reading, programming, and computing techniques. Our hybrid CMOS/memristor integrated circuit includes periphery circuitry for using memristors within digital circuits and an analog mode with direct access to memristors. The platform has been designed, fabricated, and tested for both digital and analog computing concepts.

Future research directions and opportunities for using memristor-based experimental platform in energy-efficient AI applications are vast. Currently, we are using the platform to validate multiple digital logic-in-memory and analog neuromorphic concepts within two research laboratories, and we plan to make the platform available to other research groups.



# Conclusions and Future work

“The future depends on some graduate student who is deeply suspicious of everything I have said.”

---

Geoffrey HINTON

## Summary

Revisiting the research objectives laid out in the introduction, this thesis tackles two key challenges in the field of AI: energy consumption and trustworthiness. As AI becomes increasingly integrated into our daily lives, it is imperative to follow a mindful and cautious development strategy. This approach ensures the preservation of human safety, natural resources, and the environment. The primary focus of this thesis was the development of specialized integrated circuits capable of supporting low-energy AI models, particularly for edge applications in resource-constrained environments. The incorporation of Bayesian inference – an AI technique celebrated for its transparency and explainability – was a crucial aspect of our approach, as it addresses trust issues in AI. This integration promotes the creation of applications that are not only transparent but also reliable.

Inspired by the astounding energy efficiency and intelligence of the human brain, we utilized the near-memory computing architecture, facilitated by cutting-edge nanoelectronic technology. This endeavor called for an interdisciplinary approach, merging fields such as artificial intelligence, computer architecture, and emerging technologies. A key focus was the exploitation of the non-volatility and near-memory capabilities of memristors, along with their other non-ideal characteristics. The designs we created associate logic and memory, resulting in high energy efficiency, ideally suited for edge computing. The first three chapters directly served the thesis's objectives, while the fourth chapter ventured further to address the fundamental challenges associated with memristors, a promising emerging nanodevice for new computing paradigms.

**In Chapter 1,** we set out a comprehensive exploration into the potential of near-memory computing architecture for edge AI applications. We began by investigating the evolution of making efficient chip, before discussing the limitations of von Neumann Machines with respect to the new computing and energy efficiency demands. We emphasized the shift towards In/Near-Memory Computing using emerging memories, such as memristors. The core of this chapter was the creation of a basic Bayesian machine architecture leveraging memristor technology. Our project aims to make substantial contributions to nanodevice-based Bayesian inference by bringing fully developed, efficient memristor-based accelerators to reality. This chapter laid a solid foundation for developing energy-efficient Bayesian machines using memristors and set the stage for the subsequent development of two specific integrated circuits—stochastic and logarithmic Bayesian machines—discussed in Chapters 2 and 3, respectively.

**In Chapter 2,** we reported considerable progress in developing, fabricating, and measuring a Bayesian machine implemented with a stochastic computing approach in a system with distributed memristor arrays. The machine showcases its ability to perform local computations,

minimizing energy movement, leading to far superior energy efficiency compared to a conventional microcontroller unit for a gesture recognition task. We opted for a digital computing-based design, driven by the specificities of Bayesian inference and the near-memory computing approach. This decision facilitated the deployment of a simple sense amplifier for reading memristors, yielding several benefits, including supply voltage flexibility, calibration-free operation, mitigation of read disturbances, and immunity to device variation. The digital approach also allowed us to demonstrate a complete system comprising 16 small memory blocks. The machine's use of stochastic computing offers natural resilience to soft errors and radiation, making our machine well-suited for deployment in extreme environments. The results of our work on this project have been successfully published in the *Nature Electronics* journal [14]. We further explored strategies to decrease the dominant energy consumption in our machine, associated with random number generation. Our proposition involved utilizing nanodevices to locally generate high-quality random bits, which could potentially result in significant energy cost reduction. We ventured into fabricating several prototype circuits employing unstable SMTJ devices with PCSA sensing circuitry to generate random bits. The progress made in this chapter paves the way for further exploration in the field of Bayesian machines, stochastic computing, and the efficient use of nanodevices.

**In Chapter 3,** we addressed the limitations of stochastic computing in memristor-based Bayesian machines and presented logarithmic computing as an effective solution. This method enhances precision and accelerates inference operations, all while preserving the architecture and design choices of the original Bayesian machine. The Logarithmic machine, also implemented with distributed memristor arrays, inherits all the benefits of our Bayesian inference approach, including reduced data movement, explainable models, uncertainty information, and efficient training with limited data. We conducted a comparative study based on measurements from two prototype Bayesian machines we fabricated: the logarithmic and the stochastic Bayesian machines. Both machines produced accurate measurement results under a variety of supply voltages, demonstrating the high robustness and supply voltage flexibility of our Bayesian inference machines. Further energy estimation, performed using a homemade energy estimation framework, confirmed that both designs could execute a gesture recognition task using significantly less energy than a microcontroller unit. The choice between the two designs ultimately hinges on specific applications and their respective energy and accuracy constraints. Our findings underscore the potential of memristor-based near-memory Bayesian computing as a promising solution for energy-efficient machine learning systems, even when using affordable technology such as the 130-nanometer process. The results of our work on this project have been successfully presented in the DATE 2023 conference [86]. Inspired by the encouraging results of our Bayesian machine studies, as presented in Chapters 2 and 3, we were motivated to develop a larger, more competent system capable of handling real tasks on-chip. Consequently, we designed and fabricated a large-scale, multi-computing mode chip that

features both logarithmic and stochastic computing modes. This dual-mode chip allows us to harness the benefits of both computing modes. These promising developments have set a solid foundation for us to continue working on several practical tasks for future research directions.

**In Chapter 4,** we addressed the inherent challenges associated with the use of memristors. Recognizing memristors' imperfections as significant hurdles to overcome, we underscored the crucial role of experimental platforms for confronting these issues and fostering the development and testing of innovative memristor-based concepts. To this end, we introduced our multimode memristor-based prototyping platform, designed to facilitate the implementation of both analog and digital projects. Following successful design, fabrication, and testing phases, the platform emerged as a promising conduit for the exploration of new ideas, alongside the validation and optimization of reading, programming, and computing techniques. The platform has been presented in the ASP-DAC 2023 conference for the university design contest [87]. Looking to the future, the potential for using this memristor-based experimental platform in energy-efficient AI applications is vast. Currently, the platform is employed in two research laboratories, it is being utilized to validate an array of digital logic-in-memory and analog neuromorphic concepts.

**Implications of the Research.** Our research signifies additional steps on the evolving path of emerging computing paradigms using cutting-edge technologies. This aligns with the objective of our research group, "IntegNano", aimed at incorporating nanodevice technologies to address real-world issues. Our work bridges the gap between theoretical concepts, device demonstrations, and simulations, and actual integrated system demonstrations. During this thesis, we functioned as an intermediary between two research realms: the material and device world and the algorithms and computing world. Throughout our projects, we successfully incorporated devices, especially memristors, into energy-efficient systems, demonstrating the potential viability of our approaches, such as in-near memory computing with memristors, in addressing real-world problems like AI energy efficiency and trustworthiness. This propels the field of research forward, edging closer to more mature deployment in the near future.

**Limitations.** Much like an iceberg, the visible success of research is always underpinned by invisible hard work in overcoming limitations. During our research, we encountered numerous difficulties inherent to working in emerging research fields, utilizing immature technologies, design tools, and methodologies. A substantial portion of our work focused on resolving these issues. Given our young research group, it was initially challenging to develop sophisticated systems. We chose to progress incrementally, enhancing the complexity of our designs and projects in tandem with our accumulating experience.

**Reflection.** Reflecting on our journey, it is evident that our research, albeit challenging, has been immensely rewarding. It underscored the understanding that innovative solutions re-

quiring multidisciplinary knowledge is a complex and demanding task, necessitating a mix of technical expertise, advanced manufacturing techniques, and substantial resources. This is especially true for research laboratories, which often face significant constraints due to limited resources. This journey has personally transformed my perspective on developing innovative technologies and solutions, and it has been a truly enriching journey to undertake.

## Perspectives

The journey through this thesis has led to notable advancements in integrating nanoelectronic technology in energy-efficient AI demonstrator circuits. However, these achievements are not endpoints but stepping stones leading to a multitude of new research avenues, particularly in an age where energy-efficient and trustworthy AI applications are of paramount importance. Several potential research directions, outlined below, emerge from the major contributions of this thesis.

**Scaling up Memristor-based Systems:** The large-scale Bayesian machine, presented in Chapter 3, offers one of the most immediate opportunities for continued research. With its design and fabrication incorporating 143k memristors and 285k transistors, thorough testing and characterization of this machine are essential next steps. The potential of this machine to tackle real-world tasks is a fascinating aspect waiting to be explored, particularly given its dual-mode (stochastic and logarithmic) functionality. Furthermore, it is worth investigating the development of more advanced technology nodes to further reduce the energy consumption of our Bayesian machines. While the 130nm CMOS process was employed in this thesis due to its accessibility and affordability, transitioning to more advanced nodes such as 28nm or 22nm could significantly decrease energy consumption and boost the performance of memristor-based circuits. Looking further ahead, an aspiration is to develop a core-level demonstrator, integrating a RISC-V processor for standalone control and computing capabilities. This would entail augmenting the memory hierarchy, such as adding SRAMs to the system, and expanding the memory size, such as using memristors' analog capabilities, for enabling the core to handle more complex tasks. The ultimate goal would be to develop a standalone research or open-source platform, akin to the Arduino prototyping platform, replete with I/Os, near-sensors integration, and embedded solar power cells. This platform could advance research in smart devices and edge AI, serving potential application areas like edge computing, Internet of Things (IoT) devices, and wearable technology.

**Expanding Integration of Nanodevice and Nanophysics in Computing:** Our research also paves the way for the integration of other emerging nanodevices into our Bayesian machine architecture or other AI models. While memristors were primarily chosen for their in-memory and near-memory capabilities, other nanodevices might offer additional benefits or superior



performance under certain conditions. The SMTJ prototypes developed in Chapter 2 for RNG and P-bits could be the starting point for exploring these possibilities. Characterization and modeling of these devices and circuits could lead to the development of an array-level system as an experimental platform for probabilistic computing projects.

**Tackling Memristor Imperfections for analog In-Memory Computing:** The challenge of addressing device imperfections remains despite the promising potential of the In/Near-memory computing approach with memristor devices for building energy-efficient AI systems. The multimode memristor-based prototyping platform developed in Chapter 4 provides an invaluable tool for future research. It can be used to test new designs, refine existing concepts, and deepen our understanding of memristor properties and non-ideal characteristics. Moreover, it creates an opportunity to explore memristor-based Binary Neural Networks and Ternary Neural Networks, with brain-inspired synaptic plasticity feature, which could be optimized further with an enhanced understanding. We plan to extend the availability of this platform to other research groups, fostering further advancements in this exciting field.

**Addressing On-chip Learning Challenges:** The integrated circuit developed for prototyping memristor-based projects also lays the foundation for future research in developing and testing new neuromorphic concepts and algorithms, which are hardware-friendly or adopt local learning rules for experimental implementation and testing. The Markov chain Monte Carlo (MCMC) sampling machine, a separate chip we developed as part of the MCMC project [88] (not discussed in this thesis), provides an additional platform. This platform incorporates advanced analog circuitry, paving the way for addressing in-situ learning challenges. It opens up the possibility of integrating more sophisticated Bayesian techniques, such as Bayesian neural networks, into our current Bayesian AI research direction. This could potentially lead to substantial improvements in both accuracy and uncertainty quantification. Furthermore, the hybrid Memristor/FeFET arrays in the BEOL of the 22nm FDSOI process, developed in collaboration with our partners (IM2NP and CEA-Leti), present another significant opportunity. By harnessing the potential of two promising emerging devices in tandem with the energy efficiency of 22nm FDSOI transistors, we may be able to implement in-memory learning and computing in innovative ways. For instance, one device could store the weight and the other the updates, or one could store the mean and the other the standard deviation of a Gaussian distribution.

In summary, our future research directions will explore a multidimensional solution space. This will range from the technological dimension, incorporating several emerging nanodevice technologies, to the conceptual dimension, which embraces diverse approaches, learning and inference methods, and computing approaches (analog, digital, stochastic). From an architectural perspective, this includes near-memory and in-memory architectures, while from a system scale perspective, it ranges from the device to the system level (See last Figure in this

Ph.D. thesis Fig. 4.12).

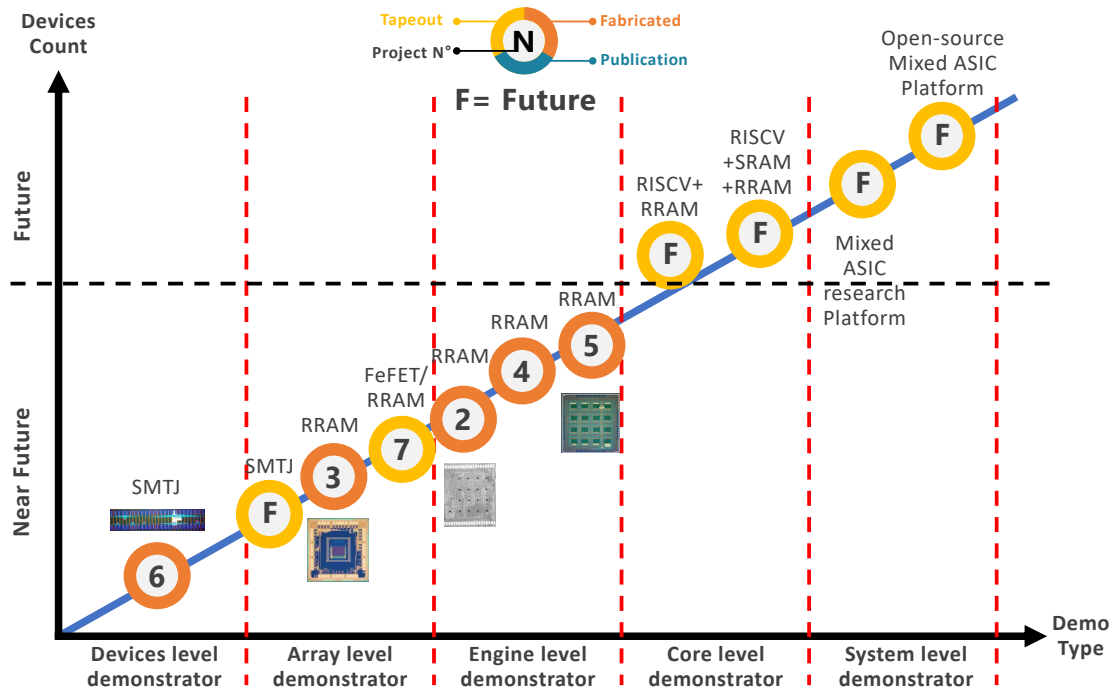


Figure 4.12: **A Perspectives plot** A forward-looking plot demonstrating the projected growth in our research efforts, with the X-axis representing the demonstrators' scale from device to full system circuit, and the Y-axis indicating devices' count in the circuits. The linear line signifies the evolution of our designs, our Moore's law, from existing to future projects, showcasing an anticipated increase in both scale and complexity.



# List of publications

## Peer-Reviewed Journal Articles

✦ **KAMEL-EDDINE HARABI**, TIFENN HIRTZLIN, CLÉMENT TURCK, ELISA VIANELLO, RAPHAËL LAURENT, JACQUES DROULEZ, PIERRE BESSIÈRE, JEAN-MICHEL PORTAL, MARC BOCQUET and DAMIEN QUERLIOZ , “A memristor-based Bayesian machine” , *Nature Electronics*, vol. 6, No. 1, p. 52–63, 2023.

[doi:10.1038/s41928-022-00886-9](https://doi.org/10.1038/s41928-022-00886-9)

✦ THOMAS DALGATY, NICCOLO CASTELLANI, CLÉMENT TURCK, **KAMEL-EDDINE HARABI**, DAMIEN QUERLIOZ and ELISA VIANELLO , “In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling” , *Nature Electronics*, vol. 4, No. 2, p. 151–161, 2021.

[doi:10.1038/s41928-020-00523-3](https://doi.org/10.1038/s41928-020-00523-3)

✦ FADI JEBALI, ATREYA MAJUMDAR, CLEMENT TURCK, **KAMEL-EDDINE HARABI**, MATHIEU-COUMBA FAYE, ELOI MUHR, JEAN-PIERRE WALDER, OLEKSANDR BILOUSOV, AMADEO MICHAUD, ELISA VIANELLO, TIFENN HIRTZLIN, FRANCOIS ANDRIEU, MARC BOCQUET, STEPHANE COLLIN, DAMIEN QUERLIOZ and JEAN-MICHEL PORTAL, “Powering AI at the Edge: A Robust, Memristor-based Binarized Neural Network with Near-Memory Computing and Miniaturized Solar Cell” , *Under review at Nature Communications* .

[doi:10.48550/arXiv.2305.12875](https://doi.org/10.48550/arXiv.2305.12875)

## Peer-Reviewed Conference Proceedings

✦ **KAMEL-EDDINE HARABI**, TURCK CLEMENT, DROUHIN MARIE, RENAUDINEAU ADRIEN, BERSANI-VERONI T, HIRTZLIN T, VIANELLO E, BOCQUET M, PORTAL J-M and QUERLIOZ D, “A Multimode Hybrid Memristor-CMOS Prototyping Platform Supporting Digital and Analog Projects” , *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, p. 184–185, 2023.

[doi:10.1145/3566097.3567944](https://doi.org/10.1145/3566097.3567944)

✦ TURCK CLEMENT, **KAMEL-EDDINE HARABI**, DROUHIN MARIE, RENAUDINEAU ADRIEN, BERSANI-VERONI T, HIRTZLIN T, VIANELLO E, BOCQUET M, PORTAL J-M and QUERLIOZ D, “Energy-Efficient Bayesian Inference Using Near-Memory Computation with Memristors” , *Proceedings of the Design, Automation and Test in Europe Conference 2023*, 2023. *Co-First author*.

[doi:10.23919/DATE56975.2023.10137312](https://doi.org/10.23919/DATE56975.2023.10137312)

✠ A. RENAUDINEAU, **KAMEL-EDDINE HARABI**, C. TURCK1, A. LABORIEUX, E. VIANELLO, M. BOCQUET, J.-M. PORTAL and D. QUERLIOZ, “Experimental Demonstration of Memristor Delay-Based Logic In-Memory Ternary Neural Network” , *Silicon Nanoelectronics Workshop 2023 (SNW 2023)*, *Co-First author*.

[doi:10.23919/SNW57900.2023.10183957](https://doi.org/10.23919/SNW57900.2023.10183957)

✠ MARIE DROUHIN, CLÉMENT TURCK, **KAMEL-EDDINE HARABI**, ADRIEN RENAUDINEAU, THOMAS BERSANI-VERONI, ELISA VIANELLO, JEAN-MICHEL PORTAL, JULIE GROLLIER, and DAMIEN QUERLIOZ, “Nanoelectronic implementation of Equilibrium Propagation” , *To be presented at Emerging Topics in Artificial Intelligence (ETAI) 2023*.

✠ HIRTZLIN TIFENN, DALGATY THOMAS, BOCQUET MARC, PORTAL JEAN-MICHEL, KLEIN JACQUES-OLIVIER, TURCK CLEMENT, **KAMEL-EDDINE HARABI**, QUERLIOZ DAMIEN and VIANELLO ELISA, “From resilience to Resistive Memory variability in Binarized Neural Networks to exploitation of variability in Bayesian Neural Network” , *2022 International Conference on IC Design and Technology (ICICDT)*, 2022.

[doi:10.1109/ICICDT56182.2022.9933076](https://doi.org/10.1109/ICICDT56182.2022.9933076)

## Conferences Without Proceedings

✠ **KAMEL-EDDINE HARABI**, JEAN-MICHEL PORTAL, JACQUES-OLIVIER KLEIN and DAMIEN QUERLIOZ, “Energy Efficient Memristor-Based Artificial Intelligence Accelerators using In/Near Memory Computing” , *ASPDAC 2023 PhD Forum, Poster presentation. Best Poster Award*.

✠ **KAMEL-EDDINE HARABI**, JEAN-MICHEL PORTAL, JACQUES-OLIVIER KLEIN and DAMIEN QUERLIOZ, “Energy Efficient Memristor-Based Artificial Intelligence Accelerators using In/Near Memory Computing” , *DATE 2023 PhD Forum, Poster presentation*.

✠ **KAMEL-EDDINE HARABI**, TIFENN HIRTZLIN, CLÉMENT TURCK, ELISA VIANELLO, RAPHAËL LAURENT, JACQUES DROULEZ, PIERRE BESSIÈRE, JEAN-MICHEL PORTAL, MARC BOCQUET and DAMIEN QUERLIOZ , “A memristor-based Stochastic Bayesian machine” , *FETCH 2022 Workshop, Poster presentation*.

# Bibliography

- [1] Dario Amodei et al. Ai and compute, May 2018.
- [2] Andrew Lohn and Micha Musser. Ai and compute: How much longer can computing power drive artificial intelligence progress. *Center for Security and Emerging Technology*. <https://doi.org/10.51593>, 2022.
- [3] Dario Amodei et al. The cost of training machines is becoming a problem, Jun 2020.
- [4] Amir Gholami. Ai and memory wall, Mar 2021.
- [5] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020.
- [6] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. Deep learning's diminishing returns: The cost of improvement is becoming unsustainable. *Ieee Spectrum*, 58(10):50–55, 2021.
- [7] Karl Rupp. Microprocessor trend data, Feb 2018.
- [8] John Park. This is not your fathers advanced semiconductor packaging...an eda perspective, Nov 2022.
- [9] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. *ACM SIGARCH computer architecture news*, 44(3):367–379, 2016.
- [10] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.
- [11] Abu Sebastian, Manuel Le Gallo, Riduan Khaddam-Aljameh, and Evangelos Eleftheriou. Memory devices and applications for in-memory computing. *Nature nanotechnology*, 15(7):529–544, 2020.

- 
- [12] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.
- [13] Seungchul Jung, Hyungwoo Lee, Sungmeen Myung, Hyunsoo Kim, Seung Keun Yoon, Soon-Wan Kwon, Yongmin Ju, Minje Kim, Wooseok Yi, Shinhee Han, et al. A cross-bar array of magnetoresistive memory devices for in-memory computing. *Nature*, 601(7892):211–216, 2022.
- [14] K-E. Harabi et al. A memristor-based bayesian machine. *Nature electronics*, 6:52–63, 2023.
- [15] Komal Chauhan. Asic design flow in vlsi engineering services – a quick guide, 2020.
- [16] Jean-Michel Muller, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, Serge Torres, et al. *Handbook of floating-point arithmetic*, volume 1. Springer, 2018.
- [17] Tifenn Hirtzlin. *Digital Implementation of Neuromorphic systems using Emerging Memory devices*. Theses, Université Paris-Saclay, November 2020.
- [18] Kerem Y Camsari, Brian M Sutton, and Supriyo Datta. P-bits for probabilistic spin logic. *Applied Physics Reviews*, 6(1):011305, 2019.
- [19] Damir Vodenicarevic, Nicolas Locatelli, Alice Mizrahi, Joseph S Friedman, Adrien F Vincent, Miguel Romera, Akio Fukushima, Kay Yakushiji, Hitoshi Kubota, Shinji Yuasa, et al. Low-energy truly random number generation with superparamagnetic tunnel junctions for unconventional computing. *Phys. Rev. Appl.*, 8(5):054045, 2017.
- [20] Shuvro Chowdhury, Andrea Grimaldi, Navid Anjum Aadit, Shaila Niazi, Masoud Mohseni, Shun Kanai, Hideo Ohno, Shunsuke Fukami, Luke Theogarajan, Giovanni Finocchio, et al. A full-stack view of probabilistic computing with p-bits: devices, architectures and algorithms. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 2023.
- [21] Fei Yu, Lixiang Li, Qiang Tang, Shuo Cai, Yun Song, and Quan Xu. A survey on true random number generators based on chaos. *Discrete Dynamics in Nature and Society*, 2019:1–10, 2019.
- [22] D. Garbin, E. Vianello, O. Bichler, Q. Rafhay, C. Gamrat, G. Ghibaudo, B. DeSalvo, and L. Perniola. Hfo<sub>2</sub>-based oxram devices as synapses for convolutional neural networks. *IEEE Transactions on Electron Devices*, 62(8):2494–2501, Aug 2015.

- 
- [23] A. Majumdar et al. Model of the weak reset process in hfo x resistive memory for deep learning frameworks. *IEEE Trans. Elect. Dev.*, 68:4925, 2021.
- [24] Thomas Dalgaty, Eduardo Esmanhotto, Niccolo Castellani, Damien Querlioz, and Elisa Vianello. Ex situ transfer of bayesian neural networks to resistive memory-based inference hardware. *Advanced Intelligent Systems*, 3(8):2000103, 2021.
- [25] Eduardo Esmanhotto, Tifenn Hirtzlin, Djohan Bonnet, Niccolo Castellani, Jean-Michel Portal, Damien Querlioz, and Elisa Vianello. Experimental demonstration of multilevel resistive random access memory programming for up to two months stable neural networks inference accuracy. *Advanced Intelligent Systems*, 4(11):2200145, 2022.
- [26] Tifenn Hirtzlin, Marc Bocquet, Bogdan Penkovsky, Jacques-Olivier Klein, Etienne Nowak, Elisa Vianello, Jean-Michel Portal, and Damien Querlioz. Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays. *Frontiers in neuroscience*, 13:1383, 2020.
- [27] Valérie Masson-Delmotte, Panmao Zhai, Hans-Otto Pörtner, Debra Roberts, Jim Skea, Priyadarshi R Shukla, Anna Pirani, Wilfran Moufouma-Okia, Clotilde Péan, Roz Pidcock, et al. Global warming of 1.5 c. *An IPCC Special Report on the impacts of global warming of*, 1(5):43–50, 2018.
- [28] Lea Berrang-Ford, James D Ford, and Jaclyn Paterson. Are we adapting to climate change? *Global environmental change*, 21(1):25–33, 2011.
- [29] David Franklin Beck. Technology development life cycle processes. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2013.
- [30] Les Robinson. A summary of diffusion of innovations, 2009.
- [31] Frank W Geels. Technological transitions as evolutionary reconfiguration processes: a multi-level perspective and a case-study. *Research policy*, 31(8-9):1257–1274, 2002.
- [32] Andrew Ng. Artificial intelligence is the new electricity. In *presentation at the Stanford MSx Future Forum*, 2017.
- [33] Michael Haenlein and Andreas Kaplan. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California management review*, 61(4):5–14, 2019.
- [34] Andreas Kaplan and Michael Haenlein. Siri, siri, in my hand: Who’s the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. *Business horizons*, 62(1):15–25, 2019.
- [35] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.



- 
- [36] John McCarthy. Programs with common sense, 1959.
- [37] Allen Newell, John C Shaw, and Herbert A Simon. Report on a general problem solving program. In *IFIP congress*, volume 256, page 64. Pittsburgh, PA, 1959.
- [38] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press, 2019.
- [39] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.
- [40] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.
- [41] Fred Attneave and MB. The organization of behavior; a neuropsychological theory, 1950.
- [42] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [43] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386, 2020.
- [44] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.
- [45] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [46] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
- [47] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.
- [48] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- [49] Carver A Mead and Misha A Mahowald. A silicon model of early visual processing. *Neural networks*, 1(1):91–97, 1988.
- [50] Carver Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990.
- [51] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

- 
- [52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [53] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.
- [54] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.
- [55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017.
- [56] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [57] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.
- [58] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- [59] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [60] Spyros Makridakis. The forthcoming artificial intelligence (ai) revolution: Its impact on society and firms. *Futures*, 90:46–60, 2017.
- [61] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.
- [62] Robert Culkin and Sanjiv R Das. Machine learning in finance: the case of deep learning for option pricing. *Journal of Investment Management*, 15(4):92–100, 2017.

- 
- [63] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.
- [64] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.
- [65] Youngeun Kwon and Minsoo Rhu. Beyond the memory wall: A case for memory-centric hpc system for deep learning. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 148–161. IEEE, 2018.
- [66] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.
- [67] Nicola Jones et al. How to stop data centres from gobbling up the world’s electricity. *Nature*, 561(7722):163–166, 2018.
- [68] Giacomo Indiveri and Shih-Chii Liu. Memory and information processing in neuromorphic systems. *Proc. IEEE*, 103(8):1379–1397, 2015.
- [69] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- [70] Keng Siau and Weiyu Wang. Building trust in artificial intelligence, machine learning, and robotics. *Cutter business technology journal*, 31(2):47–53, 2018.
- [71] Editorial. Big data needs a hardware revolution. *Nature*, 554(7691):145, February 2018.
- [72] Deyan Chen and Hong Zhao. Data security and privacy protection issues in cloud computing. In *2012 international conference on computer science and electronics engineering*, volume 1, pages 647–651. IEEE, 2012.
- [73] Zhifeng Xiao and Yang Xiao. Security and privacy in cloud computing. *IEEE communications surveys & tutorials*, 15(2):843–859, 2012.
- [74] Hassan Takabi, James BD Joshi, and Gail-Joon Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, 8(6):24–31, 2010.
- [75] David Chen, Sijia Liu, Paul Kingsbury, Sunghwan Sohn, Curtis B Storlie, Elizabeth B Habermann, James M Naessens, David W Larson, and Hongfang Liu. Deep learning and alternative learning strategies for retrospective real-world clinical data. *NPJ digital medicine*, 2(1):1–5, 2019.
- [76] Marzyeh Ghassemi, Tristan Naumann, Peter Schulam, Andrew L Beam, Irene Y Chen, and Rajesh Ranganath. A review of challenges and opportunities in machine learning for health. *AMIA Summits on Translational Science Proceedings*, 2020:191, 2020.

- 
- [77] Arun Rai. Explainable ai: From black box to glass box. *Journal of the Academy of Marketing Science*, 48(1):137–141, 2020.
- [78] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.
- [79] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.
- [80] Davide Castelvechi. Ai pioneer: 'the dangers of abuse are very real'. *Nature*, 2019.
- [81] Will Douglas. Geoffrey hinton tells us why he's now scared of the tech he helped build, MaI 2023.
- [82] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.
- [83] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.
- [84] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.
- [85] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.
- [86] C Turck, K-E Harabi, T Hirtzlin, E Vianello, R Laurent, J Droulez, P Bessiere, M Bocquet, J-M Portal, and D Querlioz. Energy-efficient bayesian inference using near-memory computation with memristors. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–2. IEEE, 2023.
- [87] K-E Harabi, Clement Turck, Marie Drouhin, Adrien Renaudineau, T Bersani-Veroni, D Querlioz, T Hirtzlin, E Vianello, M Bocquet, and J-M Portal. A multimode hybrid memristor-cmos prototyping platform supporting digital and analog projects. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, pages 184–185, 2023.

- 
- [88] Thomas Dalgaty, Niccolo Castellani, Clément Turck, Kamel-Eddine Harabi, Damien Querlioz, and Elisa Vianello. In situ learning using intrinsic memristor variability via markov chain monte carlo sampling. *Nature Electronics*, 4(2):151–161, 2021.
- [89] et al F. Jebali. Powering ai at the edge: A robust, memristor-based binarized neural network with near-memory computing and miniaturized solar cell. *Under review, Nature Communications*, :-, 2023.
- [90] A Renaudineau, K-E Harabi, C Turck, A Laborieux, E Vianello, M Bocquet, J-M Portal, and D Querlioz. Experimental demonstration of memristor delay-based logic in-memory ternary neural network. In *2023 Silicon Nanoelectronics Workshop (SNW)*, pages 43–44. IEEE, 2023.
- [91] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [92] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
- [93] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.
- [94] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- [95] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.
- [96] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.
- [97] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.
- [98] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- [99] Stephen Cass. Taking ai to the edge: Google’s tpu now comes in a maker-friendly package. *IEEE Spectrum*, 56(5):16–17, 2019.

- 
- [100] Sparsh Mittal. A survey on optimized implementation of deep learning models on the nvidia jetson platform. *Journal of Systems Architecture*, 97:428–442, 2019.
- [101] Mircea Horea Ionica and David Gregg. The movidius myriad architecture's potential for scientific computing. *IEEE Micro*, 35(1):6–14, 2015.
- [102] Danijela Marković, Alice Mizrahi, Damien Querlioz, and Julie Grollier. Physics for neuro-morphic computing. *Nature Reviews Physics*, 2(9):499–510, 2020.
- [103] Shimeng Yu. Neuro-inspired computing with emerging nonvolatile memories. *Proc. IEEE*, 106(2):260–285, 2018.
- [104] Geoffrey W Burr, Robert M Shelby, Abu Sebastian, Sangbum Kim, Seyoung Kim, Severin Sidler, Kumar Virwani, Masatoshi Ishii, Pritish Narayanan, Alessandro Fumarola, et al. Neuromorphic computing using non-volatile memory. *Advances in Physics: X*, 2(1):89–124, 2017.
- [105] Janine M Benyus. *Biomimicry: Innovation inspired by nature*, 1997.
- [106] Kevin M Passino. *Biomimicry for optimization, control, and automation*. Springer Science & Business Media, 2005.
- [107] Michael Pawlyn. *Biomimicry in architecture*. Routledge, 2019.
- [108] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.
- [109] Michael Riordan, Lillian Hoddeson, and Conyers Herring. The invention of the transistor. *Reviews of Modern Physics*, 71(2):S336, 1999.
- [110] John Bardeen and Walter Hauser Brattain. The transistor, a semi-conductor triode. *Physical Review*, 74(2):230, 1948.
- [111] William F Brinkman, Douglas E Haggan, and William W Troutman. A history of the invention of the transistor and where it will lead us. *IEEE journal of solid-state circuits*, 32(12):1858–1865, 1997.
- [112] Charles Kittel. *Introduction to solid state physics eighth edition*. 2021.
- [113] Chris A Mack. Fifty years of moore's law. *IEEE Transactions on semiconductor manufacturing*, 24(2):202–207, 2011.
- [114] Scott E Thompson, Robert S Chau, Tahir Ghani, Kaizad Mistry, Sunit Tyagi, and Mark T Bohr. In search of " forever," continued transistor scaling one new material at a time. *IEEE Transactions on semiconductor manufacturing*, 18(1):26–36, 2005.

- 
- [115] Simon M Sze, Yiming Li, and Kwok K Ng. *Physics of semiconductor devices*. John Wiley & Sons, 2021.
- [116] JA Hoerni. Patents [method of manufacturing semiconductor devices-us patent no. 3,025,589]. *IEEE Solid-State Circuits Society Newsletter*, 12(2):41–42, 2007.
- [117] Scott E Thompson and Srivatsan Parthasarathy. Moore’s law: the future of si microelectronics. *Materials today*, 9(6):20–25, 2006.
- [118] Daniel Pham, Larry Larson, and Ji-Woon Yang. Finfet device junction formation challenges. In *2006 International Workshop on Junction Technology*, pages 73–77. IEEE, 2006.
- [119] Jean-Pierre Colinge et al. *FinFETs and other multi-gate transistors*, volume 73. Springer, 2008.
- [120] TSMC. Tsmc holds 3nm volume production and capacity expansion ceremony, marking a key milestone for advanced manufacturing, 12 2022.
- [121] Sorin Cristoloveanu. Silicon on insulator technologies and devices: from present to future. *Solid-State Electronics*, 45(8):1403–1411, 2001.
- [122] R Carter, J Mazurier, L Pirro, JU Sachse, P Baars, J Faul, C Grass, G Grasshoff, P Javorka, T Kammler, et al. 22nm fdsoi technology for emerging mobile, internet-of-things, and rf applications. In *2016 IEEE International Electron Devices Meeting (IEDM)*, pages 2–2. IEEE, 2016.
- [123] Krutideepa Bhol, Biswajit Jena, and Umakanta Nanda. Journey of mosfet from planar to gate all around: A review. *Recent Patents on Nanotechnology*, 16(4):326–332, 2022.
- [124] Samsung. Optimized 3nm process achieves 45% reduced power usage, 23% improved performance and 16% smaller surface area compared to 5nm process, June 2022.
- [125] McKinsey & Company. Semiconductor design and manufacturing: Achieving leading-edge capabilities, August 2020.
- [126] McKinsey & Company. The chips and science act: Here’s what’s in it, October 2022.
- [127] Richard Waters. Can intel become the chip champion the us needs?, APRIL 2023.
- [128] C. Textor. Number of undergraduate engineering graduates from chinese higher education institutions from 2011 to 2021, Feb 2023.
- [129] Chun-Chao Lin. Taiwan shows how winning the semiconductor race takes more than money, October 2022.
- [130] M Mitchell Waldrop. The chips are down for moore’s law. *Nature News*, 530(7589):144, 2016.

- 
- [131] MARK LAPEDUS. Transistors reach tipping point at 3nm, FEBRUARY 2022.
- [132] Ratneshwar K Ratnesh, A Goel, G Kaushik, H Garg, M Singh, B Prasad, et al. Advancement and challenges in mosfet scaling. *Materials Science in Semiconductor Processing*, 134:106002, 2021.
- [133] Nicholas R Glavin, Rahul Rao, Vikas Varshney, Elisabeth Bianco, Amey Apte, Ajit Roy, Emilie Ringe, and Pulickel M Ajayan. Emerging applications of elemental 2d materials. *Advanced Materials*, 32(7):1904302, 2020.
- [134] Bhavin J Shastri, Alexander N Tait, Thomas Ferreira de Lima, Wolfram HP Pernice, Harish Bhaskaran, C David Wright, and Paul R Prucnal. Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics*, 15(2):102–114, 2021.
- [135] Laszlo Gyongyosi and Sandor Imre. A survey on quantum computing technology. *Computer Science Review*, 31:51–71, 2019.
- [136] Wayne M Moreau. *Semiconductor lithography: principles, practices, and materials*. Springer Science & Business Media, 2012.
- [137] John H Bruning. Optical lithography: 40 years and holding. In *Optical Microlithography XX*, volume 6520, pages 62–74. SPIE, 2007.
- [138] Kanti Jain. Excimer laser lithography. 1990.
- [139] Marc D Levenson, NS Viswanathan, and Robert A Simpson. Improving resolution in photolithography with a phase-shifting mask. *IEEE Transactions on electron devices*, 29(12):1828–1836, 1982.
- [140] Oberdan W Otto, Joseph G Garofalo, KK Low, Chi-Min Yuan, Richard C Henderson, Christophe Pierrat, Robert L Kostelak, Sheila Vaidya, and PK Vasudev. Automated optical proximity correction: a rules-based approach. In *Optical/Laser Microlithography VII*, volume 2197, pages 278–293. SPIE, 1994.
- [141] Soichi Owa and Hiroyuki Nagasaka. Immersion lithography: its potential performance and issues. In *Optical Microlithography XVI*, volume 5040, pages 724–733. SPIE, 2003.
- [142] Vivek Bakshi. Euv lithography. 2009.
- [143] Joseph M Steigerwald, Shyam P Murarka, and Ronald J Gutmann. *Chemical mechanical planarization of microelectronic materials*. John Wiley & Sons, 1997.
- [144] Steven M George. Atomic layer deposition: an overview. *Chemical reviews*, 110(1):111–131, 2010.



- 
- [145] John Robertson and Robert M Wallace. High-k materials and metal gates for cmos applications. *Materials Science and Engineering: R: Reports*, 88:1–41, 2015.
- [146] Laung-Terng Wang, Yao-Wen Chang, and Kwang-Ting Tim Cheng. *Electronic design automation: synthesis, verification, and test*. Morgan Kaufmann, 2009.
- [147] Mario R Barbacci. Instruction set processor specifications (isps): The notation and its applications. *IEEE Transactions on Computers*, 100(1):24–40, 1981.
- [148] Laurence Nagel and Donald O Pederson. Spice (simulation program with integrated circuit emphasis). 1973.
- [149] Pong P Chu. *RTL hardware design using VHDL: coding for efficiency, portability, and scalability*. John Wiley & Sons, 2006.
- [150] Brucek Khailany. Accelerating chip design with machine learning. In *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, pages 33–33, 2020.
- [151] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.
- [152] Scott McCartney. *ENIAC: The triumphs and tragedies of the world's first computer*. Walker & Company, 1999.
- [153] John Von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993.
- [154] John W Backus. The ibm 701 speedcoding system. *Journal of the ACM (JACM)*, 1(1):4–6, 1954.
- [155] Gene M Amdahl, Gerrit A Blaauw, and Frederick P Brooks. Architecture of the ibm system/360. *IBM Journal of Research and Development*, 8(2):87–101, 1964.
- [156] Robert Noyce and Marcian Hoff. A history of microprocessor development at intel. *IEEE Micro*, 1(01):8–21, 1981.
- [157] Jeff Dean, David Patterson, and Cliff Young. A new golden age in computer architecture: Empowering the machine-learning revolution. *IEEE Micro*, 38(2):21–29, 2018.
- [158] John D Owens, Mike Houston, David Luebke, Simon Green, John E Stone, and James C Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.
- [159] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proc. ISCA*, pages 1–12. IEEE, 2017.

- 
- [160] Rao R Tummala. Packaging: past, present and future. In *2005 6th International Conference on Electronic Packaging Technology*, pages 3–7. IEEE, 2005.
- [161] Austin Lancaster and Manish Keswani. Integrated circuit packaging review with an emphasis on 3d packaging. *Integration*, 60:204–212, 2018.
- [162] Ardavan Pedram, Stephen Richardson, Mark Horowitz, Sameh Galal, and Shahar Kvatinsky. Dark memory and accelerator-rich system optimization in the dark silicon era. *IEEE Design & Test*, 34(2):39–50, 2017.
- [163] Damien Querlioz, Olivier Bichler, Adrien Francis Vincent, and Christian Gamrat. Bioinspired programming of memory devices for implementing an inference engine. *Proc. IEEE*, 103(8):1398–1416, 2015.
- [164] Carver Mead and Mohammed Ismail. *Analog VLSI implementation of neural systems*, volume 80. Springer Science & Business Media, 1989.
- [165] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.
- [166] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.
- [167] Naveen Verma, Hongyang Jia, Hossein Valavi, Yinqi Tang, Murat Ozatay, Lung-Yen Chen, Bonan Zhang, and Peter Deaville. In-memory computing: Advances and prospects. *IEEE Solid-State Circuits Magazine*, 11(3):43–55, 2019.
- [168] Daniele Ielmini and H-S Philip Wong. In-memory computing with resistive switching devices. *Nature Electronics*, 1(6):333, 2018.
- [169] Gagandeep Singh, Lorenzo Chelini, Stefano Corda, Ahsan Javed Awan, Sander Stuijk, Roel Jordans, Henk Corporaal, and Albert-Jan Boonstra. Near-memory computing: Past, present, and future. *Microprocessors and Microsystems*, 71:102868, 2019.
- [170] Hongshin Jun, Jinhee Cho, Kangseol Lee, Ho-Young Son, Kwiwook Kim, Hanho Jin, and Keith Kim. Hbm (high bandwidth memory) dram technology and architecture. In *2017 IEEE International Memory Workshop (IMW)*, pages 1–4. IEEE, 2017.
- [171] Geoffrey W Burr and Paul Franzon. Storage class memory. *Emerging Nanoelectronic Devices*, pages 498–510, 2014.

- 
- [172] Sabina Spiga, Abu Sebastian, Damien Querlioz, and Bipin Rajendran. *Memristive Devices for Brain-Inspired Computing: From Materials, Devices, and Circuits to Applications-Computational Memory, Deep Learning, and Spiking Neural Networks*. Woodhead Publishing, 2020.
- [173] Leon Chua. Memristor-the missing circuit element. *IEEE Transactions on circuit theory*, 18(5):507–519, 1971.
- [174] Dmitri B Strukov, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. The missing memristor found. *nature*, 453(7191):80–83, 2008.
- [175] H-S Philip Wong, Heng-Yuan Lee, Shimeng Yu, Yu-Sheng Chen, Yi Wu, Pang-Shiu Chen, Byoungil Lee, Frederick T Chen, and Ming-Jinn Tsai. Metal–oxide rram. *Proceedings of the IEEE*, 100(6):1951–1970, 2012.
- [176] Bernard Dieny, Ioan Lucian Prejbeanu, Kevin Garello, Pietro Gambardella, Paulo Freitas, Ronald Lehndorff, Wolfgang Raberg, Ursula Ebels, Sergej O Demokritov, Johan Akerman, et al. Opportunities and challenges for spintronics in the microelectronics industry. *Nature Electronics*, 3(8):446–459, 2020.
- [177] H-S Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P Reifenberg, Bipin Rajendran, Mehdi Asheghi, and Kenneth E Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12):2201–2227, 2010.
- [178] Thomas Mikolajick, Stefan Slesazek, Min Hyuk Park, and Uwe Schroeder. Ferroelectric hafnium oxide for ferroelectric random-access memories and ferroelectric field-effect transistors. *Mrs Bulletin*, 43(5):340–346, 2018.
- [179] T Ali, P Polakowski, S Riedel, T Büttner, T Kämpfe, M Rudolph, B Pätzold, K Seidel, D Löhr, R Hoffmann, et al. Silicon doped hafnium oxide (hso) and hafnium zirconium oxide (hzo) based fefet: A material relation to device physics. *Applied Physics Letters*, 112(22):222903, 2018.
- [180] JG Alzate, Umut Arslan, Peng Bai, Justin Brockman, Yu-Jin Chen, Nilanjan Das, Kevin Fischer, Tahir Ghani, Philip Heil, Patrick Hentges, et al. 2 mb array-level demonstration of stt-mram process and performance towards l4 cache applications. In *2019 IEEE International Electron Devices Meeting (IEDM)*, pages 2–4. IEEE, 2019.
- [181] Shahar Kvatinsky, Dmitry Belousov, Slavik Liman, Guy Satat, Nimrod Wald, Eby G Friedman, Avinoam Kolodny, and Uri C Weiser. Magic—memristor-aided logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(11):895–899, 2014.
- [182] Hussein Nili, Gina C Adam, Brian Hoskins, Mirko Prezioso, Jeeseon Kim, M Reza Mahmoodi, Farnood Merrikh Bayat, Omid Kavehei, and Dmitri B Strukov. Hardware-intrinsic

- security primitives enabled by analogue state and nonlinear conductance variations in integrated memristors. *Nature Electronics*, 1(3):197–202, 2018.
- [183] Stefano Ambrogio, Pritish Narayanan, Hsinyu Tsai, Robert M Shelby, Irem Boybat, Carmelo Nolfo, Severin Sidler, Massimo Giordano, Martina Bodini, Nathan CP Farinha, et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, 558(7708):60, 2018.
- [184] Mirko Prezioso, Farnood Merrih-Bayat, BD Hoskins, Gina C Adam, Konstantin K Likharev, and Dmitri B Strukov. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, 521(7550):61, 2015.
- [185] Zhongrui Wang, Saumil Joshi, Sergey Savel'ev, Wenhao Song, Rivu Midya, Yunning Li, Mingyi Rao, Peng Yan, Shiva Asapu, Ye Zhuo, et al. Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electronics*, 1(2):137, 2018.
- [186] Cheng-Xin Xue, Yen-Cheng Chiu, Ta-Wei Liu, Tsung-Yuan Huang, Je-Syu Liu, Ting-Wei Chang, Hui-Yao Kao, Jing-Hong Wang, Shih-Ying Wei, Chun-Ying Lee, et al. A cmos-integrated compute-in-memory macro based on resistive random-access memory for ai edge devices. *Nature Electronics*, 4(1):81–90, 2021.
- [187] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.
- [188] Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.
- [189] Pierre Bessière, Emmanuel Mazer, Juan Manuel Ahuactzin, and Kamel Mekhnacha. *Bayesian programming*. CRC press, 2013.
- [190] Rens Van de Schoot, David Kaplan, Jaap Denissen, Jens B Asendorpf, Franz J Neyer, and Marcel AG Van Aken. A gentle introduction to bayesian analysis: Applications to developmental research. *Child development*, 85(3):842–860, 2014.
- [191] Jean Laurens and Jacques Droulez. Bayesian processing of vestibular information. *Biological cybernetics*, 96(4):389–404, 2007.
- [192] Tai Sing Lee and David Mumford. Hierarchical bayesian inference in the visual cortex. *JOSA A*, 20(7):1434–1448, 2003.
- [193] Wolfgang Maass. Noise as a resource for computation and learning in networks of spiking neurons. *Proceedings of the IEEE*, 102(5):860–880, 2014.
- [194] David C Knill and Alexandre Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*, 27(12):712–719, 2004.

- 
- [195] Sophie Deneve. Bayesian spiking neurons i: inference. *Neural computation*, 20(1):91–117, 2008.
- [196] Audrey Houillon, Pierre Bessière, and Jacques Droulez. The probabilistic cell: implementation of a probabilistic inference by the biochemical mechanisms of phototransduction. *Acta biotheoretica*, 58(2):103–120, 2010.
- [197] Rory JE Smith, Gregory Ashton, Avi Vajpeyi, and Colm Talbot. Massively parallel bayesian inference for transient gravitational-wave astronomy. *Monthly Notices of the Royal Astronomical Society*, 498(3):4492–4502, 2020.
- [198] Charles Leech, Yordan P Raykov, Emre Ozer, and Geoff V Merrett. Real-time room occupancy estimation with bayesian machine learning using a single pir sensor and microcontroller. In *2017 IEEE Sensors Applications Symposium (SAS)*, pages 1–6. IEEE, 2017.
- [199] Xinjun Lei and Yunxin Wu. Research on mechanical vibration monitoring based on wireless sensor network and sparse bayes. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):1–13, 2020.
- [200] Joao Filipe Ferreira, Jorge Lobo, and Jorge Dias. Bayesian real-time perception algorithms on gpu. *Journal of Real-Time Image Processing*, 6(3):171–186, 2011.
- [201] Sara Zermani, Catherine Dezan, Hanen Chenini, Jean-Philippe Diguët, and Reinhardt Euler. Fpga implementation of bayesian network inference for an embedded diagnosis. In *2015 IEEE Conference on Prognostics and Health Management (PHM)*, pages 1–10. IEEE, 2015.
- [202] Ruizhe Cai, Ao Ren, Ning Liu, Caiwen Ding, Luhao Wang, Xuehai Qian, Massoud Pedram, and Yanzhi Wang. Vibnn: Hardware acceleration of bayesian neural networks. *ACM SIGPLAN Notices*, 53(2):476–488, 2018.
- [203] Shuanglong Liu, Grigorios Mingas, and Christos-Savvas Bouganis. An unbiased mcmc fpga-based accelerator in the land of custom precision arithmetic. *IEEE Transactions on Computers*, 66(5):745–758, 2016.
- [204] Raphael Frisch, Raphaël Laurent, Marvin Faix, Laurent Girin, Laurent Fesquet, Augustin Lux, Jacques Droulez, Pierre Bessière, and Emmanuel Mazer. A bayesian stochastic machine for sound source localization. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2017.
- [205] Glenn G Ko, Yuji Chai, Marco Donato, Paul N Whatmough, Thierry Tambe, Rob A Ruttenbar, David Brooks, and Gu-Yeon Wei. A 3mm<sup>2</sup> programmable bayesian inference accelerator for unsupervised machine perception using parallel gibbs sampling in 16nm. In *2020 IEEE Symposium on VLSI Circuits*, pages 1–2. IEEE, 2020.

- 
- [206] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [207] Brian R Gaines. Stochastic computing systems. In *Advances in information systems science*, pages 37–172. Springer, 1969.
- [208] Armin Alaghi and John P Hayes. Survey of stochastic computing. *ACM TECS*, 12(2s):92, 2013.
- [209] Leonel Sousa. Nonconventional computer arithmetic circuits, systems and applications. *IEEE Circ. Syst. Magazine*, 21(1):6–40, 2021.
- [210] Behrooz Parhami. Computing with logarithmic number system arithmetic: Implementation methods and performance benefits. *Computers and Electrical Engineering*, 87:106800, 2020.
- [211] Riduan Khaddam-Aljameh, Milos Stanisavljevic, Jordi Fornet Mas, Geethan Karunaratne, Matthias Brändli, Feng Liu, Abhairaj Singh, Silvia M Müller, Urs Egger, Anastasios Petropoulos, et al. Hermes-core—a 1.59-tops/mm<sup>2</sup> pcm on 14-nm cmos in-memory compute core using 300-ps/lsb linearized cco-based adcs. *IEEE Journal of Solid-State Circuits*, 57(4):1027–1038, 2022.
- [212] Peng Yao, Huaqiang Wu, Bin Gao, Jianshi Tang, Qingtian Zhang, Wenqiang Zhang, Joshua Yang, and He Qian. Fully hardware-implemented memristor convolutional neural network. *Nature*, 577(7792):641–646, 2020.
- [213] Weier Wan, Rajkumar Kubendran, S Burc Eryilmaz, Wenqiang Zhang, Yan Liao, Dabin Wu, Stephen Deiss, Bin Gao, Priyanka Raina, Siddharth Joshi, et al. 33.1 a 74 tmacs/w cmos-rram neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 498–500. IEEE, 2020.
- [214] Can Li, Jim Ignowski, Xia Sheng, Rob Wessel, Bill Jaffe, Jacqui Ingemi, Cat Graves, and John Paul Strachan. Cmos-integrated nanoscale memristive crossbars for cnn and optimization acceleration. In *2020 IEEE International Memory Workshop (IMW)*, pages 1–4. IEEE, 2020.
- [215] Di Gao, Qingrong Huang, Grace Li Zhang, Xunzhao Yin, Bing Li, Ulf Schlichtmann, and Cheng Zhuo. Bayesian inference based robust computing on memristor crossbar. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 121–126. IEEE, 2021.
- [216] Rafatul Faria, Kerem Y Camsari, and Supriyo Datta. Implementing bayesian networks with embedded stochastic mram. *AIP Advances*, 8(4):045101, 2018.

- 
- [217] Prabhat Kumar Gupta and Ramdas Kumaresan. Binary multiplication with pn sequences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(4):603–606, 1988.
- [218] Yao-Feng Chang, James A O’Donnell, Tony Acosta, Roza Kotlyar, Albert Chen, Pedro A Quintero, Nathan Strutt, Oleg Golonzka, Chris Connor, and Jeff Hicks. envm rram reliability performance and modeling in 22ffl finfet technology. In *2020 IEEE International Reliability Physics Symposium (IRPS)*, pages 1–4. IEEE, 2020.
- [219] Stefano Gregori, Alessandro Cabrini, Osama Khouri, and Guido Torelli. On-chip error correcting techniques for new-generation flash memories. *Proc. IEEE*, 91(4):602–616, 2003.
- [220] M. Bocquet, T. Hirztlin, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz. In-memory and error-immune differential rram implementation of binarized deep neural networks. In *IEDM Tech. Dig.*, page 20.6.1. IEEE, 2018.
- [221] Weisheng Zhao, Claude Chappert, Virgile Javerliac, and Jean-Pierre Noziere. High speed, high stability and low power sensing amplifier for mtj/cmos hybrid logic circuits. *IEEE Transactions on Magnetics*, 45(10):3784–3787, 2009.
- [222] Weisheng Zhao, Mathieu Moreau, Erya Deng, Yue Zhang, Jean-Michel Portal, Jacques-Olivier Klein, Marc Bocquet, Hassen Aziza, Damien Deleruyelle, Christophe Muller, et al. Synchronous non-volatile logic gate design based on resistive switching memories. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(2):443–454, 2014.
- [223] Alessandro Grossi, Cristian Zambelli, Piero Olivo, Alberto Crespo-Yepes, Javier Martin-Martinez, Rosana Rodríguez, Monserrat Nafria, Eduardo Perez, and Christian Wenger. Electrical characterization and modeling of 1t-1r rram arrays with amorphous and polycrystalline hfo<sub>2</sub>. *Solid-State Electronics*, 128:187–193, 2017.
- [224] Meng-Fan Chang, Che-Wei Wu, Chia-Cheng Kuo, Shin-Jang Shen, Ku-Feng Lin, Shu-Meng Yang, Ya-Chin King, Chorng-Jung Lin, and Yu-Der Chih. A 0.5 v 4mb logic-process compatible embedded resistive ram (reram) in 65nm cmos using low-voltage current-mode sensing scheme with 45ns random read time. In *2012 IEEE International Solid-State Circuits Conference*, pages 434–436. IEEE, 2012.
- [225] Chris Winstead. Tutorial on stochastic computing. In *Stochastic Computing: Techniques and Applications*, pages 39–76. Springer, 2019.
- [226] Pete Warden and Daniel Situnayake. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O’Reilly Media, 2019.

- 
- [227] William A Borders, Ahmed Z Pervaiz, Shunsuke Fukami, Kerem Y Camsari, Hideo Ohno, and Supriyo Datta. Integer factorization using stochastic magnetic tunnel junctions. *Nature*, 573(7774):390–393, 2019.
- [228] Rekha Govindaraj, Swaroop Ghosh, and Srinivas Katkooi. Csro-based reconfigurable true random number generator using rram. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(12):2661–2670, 2018.
- [229] Akio Fukushima, Takayuki Seki, Kay Yakushiji, Hitoshi Kubota, Hiroshi Imamura, Shinji Yuasa, and Koji Ando. Spin dice: A scalable truly random number generator based on spintronics. *Applied Physics Express*, 7(8):083001, 2014.
- [230] Simone Balatti, Stefano Ambrogio, Zhongqiang Wang, and Daniele Ielmini. True random number generation by variability of resistive switching in oxide-based devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):214–221, 2015.
- [231] Stefan Petzold, SU Sharath, Jonas Lemke, Erwin Hildebrandt, Christina Trautmann, and Lambert Alff. Heavy ion radiation effects on hafnium oxide-based resistive random access memory. *IEEE Transactions on Nuclear Science*, 66(7):1715–1718, 2019.
- [232] Tommaso Toffoli. Non-conventional computers. *Encyclopedia of electrical and electronics engineering*, 14:455–471, 1998.
- [233] V. Gupta, D. Mohapatra, Sang Phill Park, A. Raghunathan, and K. Roy. IMPACT: IMPrecise adders for low-power approximate computing. In *Proc. ISLPED*, pages 409–414, 2011.
- [234] Yongtae Kim, Yong Zhang, and Peng Li. Energy Efficient Approximate Arithmetic for Error Resilient Neuromorphic Computing. *IEEE T. VLSI Syst.*, 23(11):2733–2737, November 2015.
- [235] Jie Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *Proc. ETS*, pages 1–6, 2013.
- [236] S. Venkataramani, S.T. Chakradhar, K. Roy, and A. Raghunathan. Approximate computing and the quest for computing efficiency. In *Proc. DAC*, pages 1–6, June 2015.
- [237] Sandip Ray, Yier Jin, and Arijit Raychowdhury. The changing computing paradigm with internet of things: A tutorial introduction. *IEEE Design & Test*, 33(2):76–96, 2016.
- [238] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.
- [239] Edward H. Lee, Daisuke Miyashita, Elaina Chai, Boris Murmann, and S. Simon Wong. Lognet: Energy-efficient neural networks using logarithmic computation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5900–5904, 2017.



- 
- [240] I. Kouretas and V. Paliouras. Logarithmic number system for deep learning. In *2018 7th International Conference on Modern Circuits and Systems Technologies (MOCASST)*, pages 1–4, 2018.
- [241] Sangyun Oh, Hyeonuk Sim, Sugil Lee, and Jongeun Lee. Automated log-scale quantization for low-cost deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 742–751, 2021.
- [242] Ruizhou Ding, Zeye Liu, Ting-Wu Chin, Diana Marculescu, and Ronald D Blanton. Flightnns: Lightweight quantized deep neural networks for fast and accurate inference. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.
- [243] Said Hamdioui, Shahar Kvatinsky, Gert Cauwenberghs, Lei Xie, Nimrod Wald, Siddharth Joshi, Hesham Mostafa Elsayed, Henk Corporaal, and Koen Bertels. Memristor for computing: Myth or reality? In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 722–731. IEEE, 2017.
- [244] Doo Seok Jeong, Kyung Min Kim, Sungho Kim, Byung Joon Choi, and Cheol Seong Hwang. Memristors for energy-efficient new computing paradigms. *Advanced Electronic Materials*, 2(9):1600090, 2016.
- [245] Miao Hu, Catherine E. Graves, Can Li, Yunning Li, Ning Ge, Eric Montgomery, Noraica Davila, Hao Jiang, R. Stanley Williams, J. Joshua Yang, Qiangfei Xia, and John Paul Strachan. Memristor-based analog computation and neural network classification with a dot product engine. *Advanced Materials*, 30(9):1705914, 2018.
- [246] Adnan Mehonic, Abu Sebastian, Bipin Rajendran, Osvaldo Simeone, Eleni Vasilaki, and Anthony J Kenyon. Memristors—from in-memory computing, deep learning acceleration, and spiking neural networks to the future of neuromorphic and bio-inspired computing. *Advanced Intelligent Systems*, 2(11):2000085, 2020.
- [247] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- [248] Olivier Pourret, Patrick Naïm, and Bruce Marcot. *Bayesian networks: a practical guide to applications*. John Wiley & Sons, 2008.
- [249] Weier Wan, Rajkumar Kubendran, Clemens Schaefer, Sukru Burc Eryilmaz, Wenqiang Zhang, Dabin Wu, Stephen Deiss, Priyanka Raina, He Qian, Bin Gao, et al. A compute-in-memory chip based on resistive random-access memory. *Nature*, 608(7923):504–512, 2022.
- [250] Marc Bocquet, Tifenn Hirtzlin, J-O Klein, Etienne Nowak, Elisa Vianello, J-M Portal, and Damien Querlioz. In-memory and error-immune differential rram implementation of

- binarized deep neural networks. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 20–6. IEEE, 2018.
- [251] F Jebali, Atreya Majumdar, Axel Laborieux, Tifenn Hirtzlin, Elisa Vianello, JP Walder, Marc Bocquet, Damien Querlioz, and Jean-Michel Portal. Capc: A configurable analog pop-count circuit for near-memory binary neural networks. In *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 158–161. IEEE, 2021.
- [252] Axel Laborieux, Maxence Ernout, Tifenn Hirtzlin, and Damien Querlioz. Synaptic meta-plasticity in binarized neural networks. *Nature communications*, 12(1):2549, 2021.
- [253] Axel Laborieux, Marc Bocquet, Tifenn Hirtzlin, J-O Klein, L Herrera Diez, Etienne Nowak, Elisa Vianello, J-M Portal, and Damien Querlioz. Low power in-memory implementation of ternary neural networks with resistive ram-based synapse. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 136–140. IEEE, 2020.
- [254] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.
- [255] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

# Résumé de Thèse en Français

## Introduction

L'Intelligence Artificielle (IA) se trouve à l'épicentre d'une révolution technologique, détenant un pouvoir transformateur dans de nombreux secteurs de la société. Cependant, l'ascension de l'IA apporte avec elle deux défis centraux qui nécessitent notre attention urgente : l'efficacité énergétique et la fiabilité. D'une part, les exigences énergétiques croissantes de l'industrie de l'IA, alimentées par les besoins computationnels élevés des modèles d'IA, exercent une pression sur les émissions mondiales de carbone. Cela, à son tour, pose des menaces à la durabilité environnementale et limite le déploiement de l'IA dans des contextes à ressources limitées tels que les dispositifs de bordure. D'autre part, la nature "boîte noire" de nombreux systèmes d'IA et leurs processus de prise de décision opaques suscitent des préoccupations quant à leur fiabilité. Ces problèmes représentent un obstacle considérable à l'acceptation généralisée et à l'application responsable de l'IA.

Pour répondre à ces défis, cette thèse suit une approche interdisciplinaire, chevauchant les domaines de l'intelligence artificielle, de l'architecture informatique et des technologies émergentes (Fig 1). L'objectif principal est d'exploiter le potentiel de la technologie nanoelectronique nouvelle, spécifiquement les memristors, pour soutenir des paradigmes de calcul à faible énergie pour les modèles d'IA, permettant ainsi leur mise en œuvre dans des contextes à ressources limitées. En parallèle, nous utilisons l'inférence bayésienne, une technique d'IA entièrement explicative, pour résoudre les problèmes de confiance inhérents à l'IA, favorisant ainsi le développement d'applications d'IA transparentes et fiables.

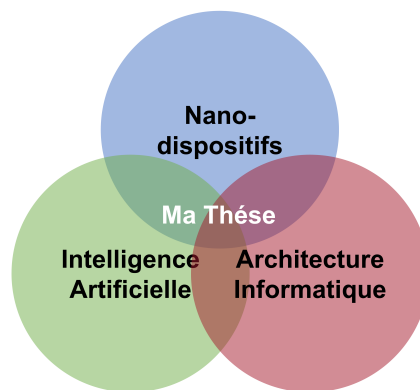


Figure 1: **Calcul Bio-inspiré.** La convergence des avancements dans les algorithmes d'IA, les architectures informatiques, et les nanodispositifs contribue à l'émergence du calcul neuromorphique, offrant des solutions potentielles aux défis prédominants en IA, tels que l'efficacité énergétique.

Cette thèse est le produit d'une collaboration complète impliquant neuf projets de recherche avec des

équipes de C2N, CEA Leti, IM2NP et Spintec (Fig 2). Ces projets ont cherché à explorer plusieurs solutions, y compris le développement de circuits intégrés, qui servent de base à nos modèles d'IA basés sur les memristors. Ces circuits spécialisés incorporent une variété de technologies nanoelectroniques émergentes telles que la RAM résistive (ReRAM), la RAM magnétorésistive (MRAM) et la RAM ferroélectrique (FeRAM).

La thèse est structurée en quatre chapitres clés. Le chapitre 1 offre un aperçu du calcul dans la mémoire proche en utilisant les memristors comme une solution viable au défi de l'efficacité énergétique en IA, tandis que le chapitre 2 approfondit la mise en œuvre d'un système bayésien stochastique basé sur des memristors. Dans le chapitre 3, nous adoptons le calcul logarithmique dans l'architecture de la machine bayésienne et ses implications pour l'efficacité énergétique. Le chapitre 4 introduit un circuit intégré conçu pour le prototypage de projets basés sur les memristors. Chaque chapitre fournit une analyse approfondie des processus de conception, de fabrication et de test et les implications des résultats pour le domaine. Ce travail de thèse souligne le potentiel de la technologie des memristors et de l'inférence bayésienne pour relever les deux défis centraux de l'IA : l'efficacité énergétique et la fiabilité.

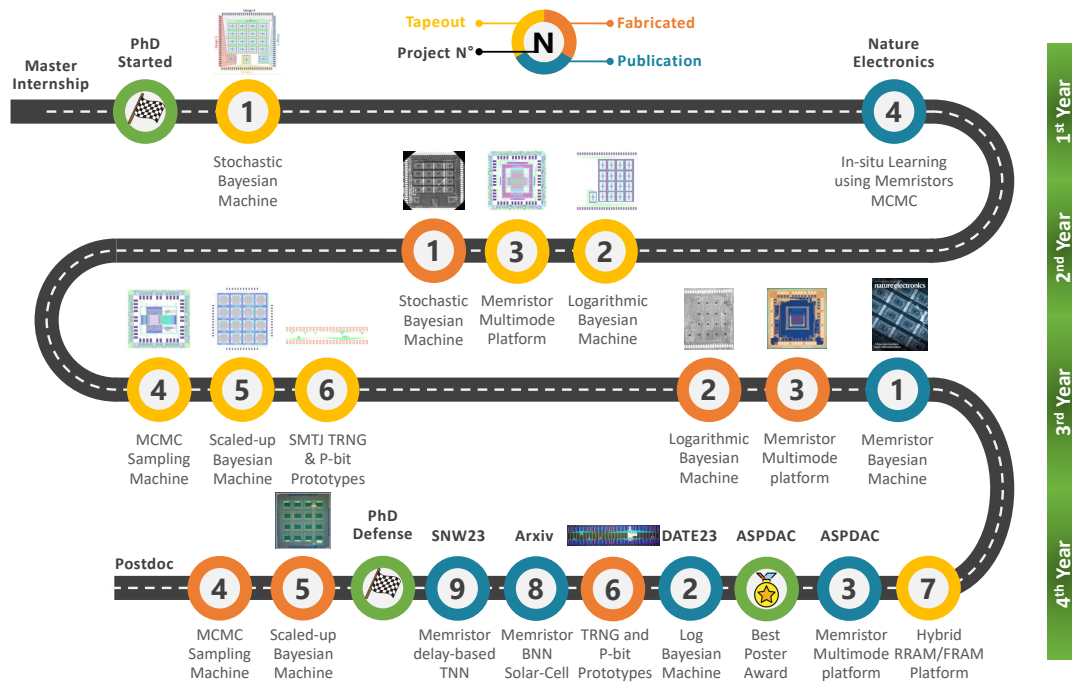


Figure 2: **Infographie de ma Thèse de Doctorat.** Durant ma thèse, j'ai été incorporé principalement ou partiellement dans neuf projets de recherche, aboutissant à six publications et à la conception de sept circuits intégrés basés sur des nanotechnologies émergentes (1, 2, 3, 4 et 5 sont basés sur la RRAM, 6 est basé sur la MRAM, et 7 sur la FRAM). Les numéros représentent les projets, le code couleur est jaune pour le design déposé (envoyé pour fabrication), orange pour les circuits fabriqués et les tests commencés, et bleu pour la publication de l'article. La plupart des designs sont fabriqués dans un processus hybride CMOS-Nanodevice de 130nm ; seul le design 7 est basé sur un processus hybride FDSOI-Nanodevice de 22nm.

## Résumé des Chapitres

### Résumé du Chapitre 1: Machine Bayésienne à base du Calcul Proche de la Mémoire

Ce premier chapitre met l'accent sur l'application du raisonnement bayésien (Eq 1) à une architecture de calcul proche de la mémoire pour l'intelligence artificielle en périphérie (Fig 3). Il décrit en détail comment construire des Machines Bayésiennes à l'aide de memristors. Une courte revue de l'évolution de la conception des puces sert de préambule à cette discussion, soulignant les motivations qui ont conduit à l'élaboration de nos Machines Bayésiennes. Cette section fournit une explication approfondie des choix de conception et du processus de développement nécessaires pour implémenter ces Machines Bayésiennes proches de la mémoire.

$$P(Y|O) = \frac{P(O|Y)P(Y)}{P(O)}. \quad (1)$$

L'ambition ultime de ce projet est de repousser les frontières de l'inférence bayésienne basée sur les nanodispositifs et de réaliser des systèmes entièrement fabriqués qui renforcent la maturité des accélérateurs bayésiens basés sur les memristors. Pour y parvenir, le projet mobilise une approche collaborative rassemblant des compétences issues de domaines tels que la théorie bayésienne, la modélisation et la caractérisation des dispositifs à memristor, ainsi que la conception de circuits intégrés.

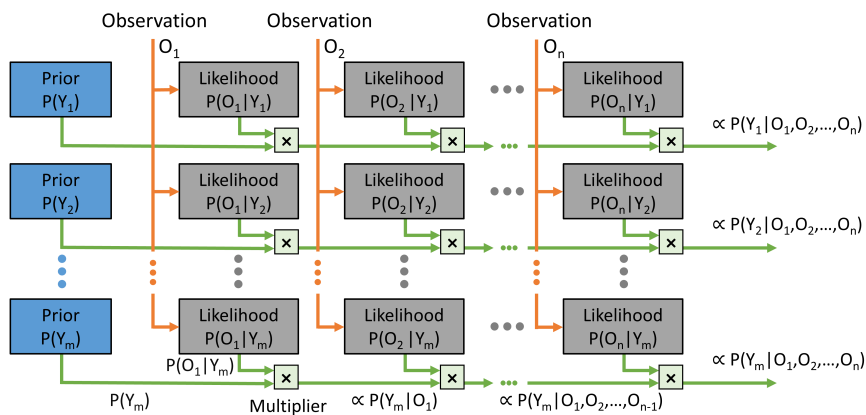


Figure 3: **Architecture générale de la machine bayésienne.** Les probabilités sont stockées dans des matrices de mémoire de probabilités implémentées par des réseaux de memristors. Les observations du monde réel sélectionnent les valeurs de probabilité appropriées à partir des matrices de mémoire de probabilités, en fonction desquelles les valeurs de probabilité sont lues à partir des matrices de probabilités, qui sont ensuite multipliées par des multiplicateurs. En sortie, les résultats générés codent la distribution a posteriori.

Le chapitre s'achève en préparant le terrain pour les discussions futures sur la Machine Bayésienne Stochastique (Chapitre 2) et la Machine Bayésienne Logarithmique (Chapitre 3). Ces circuits intégrés se

distinguent par leur flexibilité, leur simplicité et leur grande robustesse face à la variabilité des dispositifs et aux événements uniques perturbateurs. Ces caractéristiques établissent une base solide pour les recherches présentées dans cette thèse.

## Résumé du Chapitre 2: Une Machine Bayésienne Stochastique à base de Memristors

Le Chapitre 2 décrit en détail le développement et les implications potentielles d'un système bayésien stochastique innovant basé sur des memristors. Une combinaison unique de processus CMOS/memristor forme le cœur de ce système (Fig 4.d), intégrant plus de deux mille memristors et trente mille transistors dans une seule puce (Fig 4.a). Le récit débute par l'exploration de l'architecture révolutionnaire du système. Ce design exploite les avantages de la mémoire entièrement distribuée et de le calcul stochastique, réduisant de façon significative le mouvement des données en effectuant des calculs localement (Fig 5.a). De ce fait, le système bayésien dépasse de manière substantielle les implémentations traditionnelles de l'inférence bayésienne sur une unité de microcontrôleur en termes de consommation d'énergie. Ces découvertes sont étayées par une comparaison exhaustive de la consommation d'énergie par rapport à la précision dans une tâche de reconnaissance de gestes de la main.

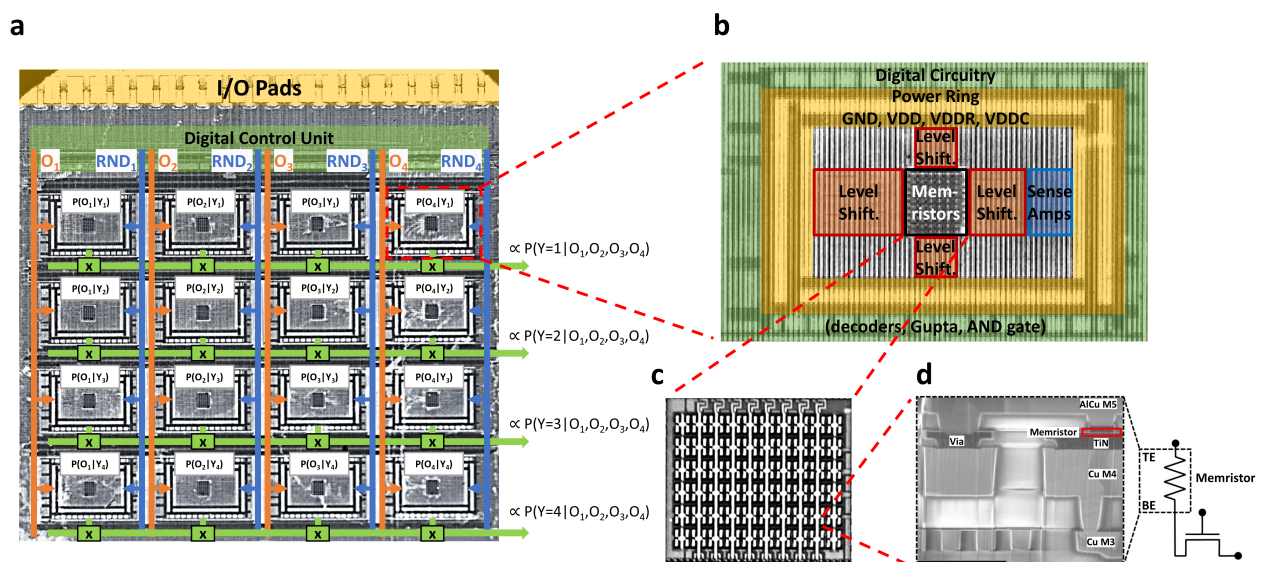


Figure 4: **La Machine bayésienne fabriquée basée sur des memristors.** **a** Photographie en microscopie optique de la puce du système bayésien. **b** Détail du bloc de vraisemblance, qui se compose de circuits numériques et du bloc de mémoire avec son circuit périphérique. **c** Photographie de la matrice de memristors 2T2R. **d** Image de microscopie électronique à balayage d'un memristor à l'arrière du processus hybride memristor/GuPTA/CMOS. Toutes les sous-figures utilisent des codes couleurs cohérents.

La performance supérieure du système repose sur son utilisation de la mémoire non volatile. Cette caractéristique offre au système bayésien une fonction marche/arrêt instantanée précieuse, validant son utilité même dans les environnements les plus extrêmes. Les décisions de conception de la machine bayésienne ont

été guidées par les exigences spécifiques de l'inférence bayésienne. Requirant une précision supérieure à ce que les memristors analogiques pourraient offrir, la conception numérique a évité le besoin d'opérations de multiplication et d'accumulation. Cette déviation par rapport aux accélérateurs de réseau neuronal basés sur des memristors courants a conduit à une conception plus flexible qui accueille de multiples petits blocs de mémoire. L'un des attributs notables de ce système bayésien est sa résilience intrinsèque aux erreurs logicielle, une conséquence directe de sa dépendance à le calcule stochastique. Cette caractéristique, couplée à la robustesse du stockage de memristors contre les radiations, rend le système bayésien idéal pour une utilisation dans des environnements difficiles.

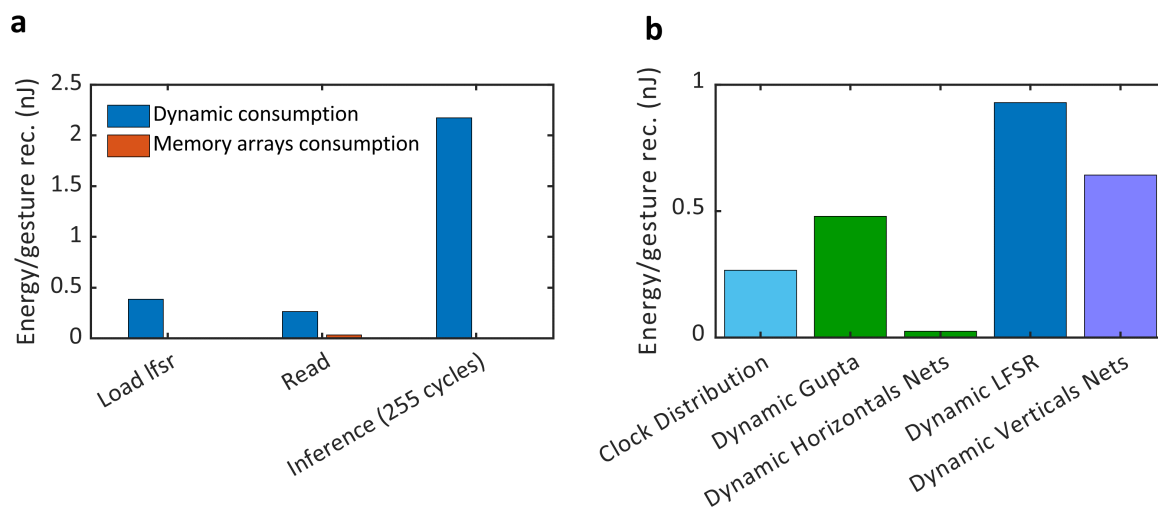


Figure 5: **Analyse de l'énergie de la machine bayésienne lors d'une tâche de reconnaissance de gestes.** **a** Consommation d'énergie du système (consommation dynamique et des block de mémoire) lors des trois phases de calcul : chargement des valeurs de départ dans le LFSR, lecture des mémoires, et l'inférence réelle de 255 cycles. **b** Consommation d'énergie des points importants du système lors de la phase d'inférence de 255 cycles. Toutes les valeurs d'énergie sont données pour une tension d'alimentation de 1,2 volt.

Bien que la performance énergétique du système soit notable, la consommation d'énergie principale a été attribuée à la génération de nombres aléatoires (Fig 5.b). Comme contre-mesure, nous avons envisagé la possibilité de générer localement des bits aléatoires en utilisant des nanodispositifs. Le développement ultérieur de circuits prototypes, utilisant des dispositifs SMTJ instables et un circuit de détection PCSA, a ouvert une voie prometteuse pour des économies d'énergie supplémentaires dans notre machine bayésienne.

Le chapitre se conclut en envisageant l'évolution future de ce projet. La poursuite d'un système doté d'une mémoire plus grande et d'une capacité de calcul supérieure, capable de gérer des tâches réelles sur la puce, est à l'horizon. Couplée à la perspective de réduire encore plus la consommation d'énergie en redimensionnant la conception à des nœuds technologiques plus avancés, l'avenir de ce système bayésien innovant semble prometteur.

### Résumé du Chapitre 3: Une Machine Bayésienne Logarithmique à base de Memristors

Ce chapitre entreprend une exploration à travers le dédale du calcul stochastique au sein des machines bayésiennes basées sur les memristors. Confronté aux limites de précision, de vitesse d'inférence et de complexité de la génération de nombres aléatoires inhérentes à ces machines, il présente une alternative significative : le calcul logarithmique. Cette alternative se distingue par sa capacité à améliorer la précision, à accélérer les opérations d'inférence, et à gérer efficacement les problèmes liés à l'underflow numérique et à la perte de précision, notamment lorsqu'il s'agit de petites probabilités ou de grands ensembles de données.

Un aspect essentiel du calcul logarithmique réside dans sa compatibilité avec l'inférence bayésienne. Cette compatibilité repose en grande partie sur le fait qu'elle transforme le calcul du produit de la distribution a priori et des vraisemblances en opérations d'addition et de soustraction élémentaires, améliorant ainsi l'efficacité et la vitesse de l'implémentation matérielle. Le récit se concentre donc sur l'intégration du calcul logarithmique dans l'architecture existante de la machine bayésienne, tout en respectant les principes de conception fondamentaux tels que l'utilisation de memristors et d'une architecture de calcul proche de la mémoire, qui sont également présents dans le modèle stochastique. La mise en œuvre pratique de ces théories et concepts est illustrée par l'introduction et le test d'un circuit intégré de machine bayésienne logarithmique récemment développé. Soumis à des tests rigoureux, ce circuit démontre le potentiel du calcul logarithmique pour améliorer l'efficacité énergétique et la précision de l'inférence bayésienne. Une validation supplémentaire de ce potentiel est obtenue grâce à des mesures comparatives sur les machines logarithmiques et stochastiques. Malgré les imperfections des memristors, ces mesures confirment la faisabilité de l'approche de la machine bayésienne (Fig 6).

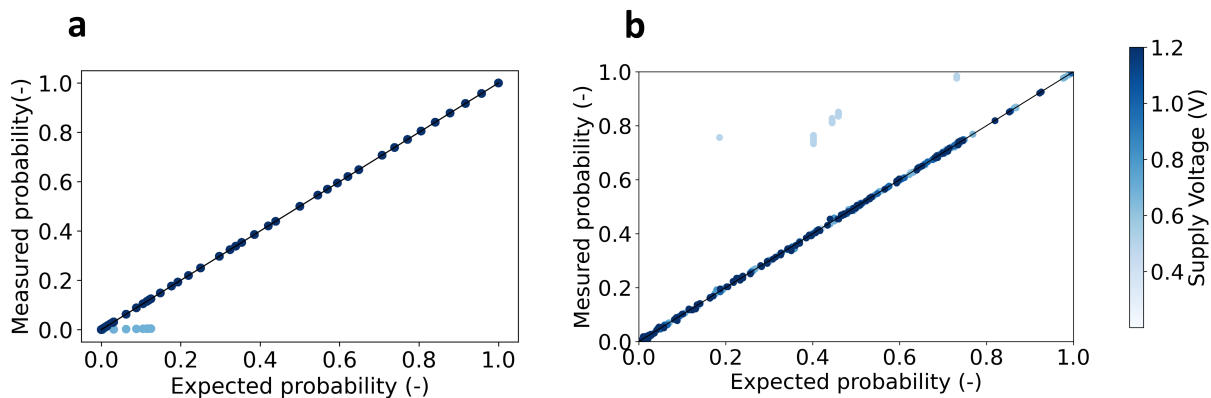


Figure 6: **Mesures d'inférence sur les machines bayésiennes fabriquées.** Sortie mesurée en fonction du résultat attendu sur la machine bayésienne fabriquée **a** logarithmique et **b** stochastique, Pour des tensions d'alimentation allant de 0,5 à 1,2 V sont superposés.

En soulignant les avantages offerts par l'approche d'inférence bayésienne, tels que les modèles explicables et la capacité à fonctionner avec des données limitées, le chapitre fait l'éloge des vertus de la machine bayésienne logarithmique. Cette machine se distingue par sa capacité à effectuer des calculs localement à l'aide de memristors distribués avec un minimum de mouvement d'énergie. Cela en fait un candidat prometteur pour gérer les situations incertaines dans les systèmes embarqués. Dans le but de résoudre le



problème de la consommation d'énergie, une étude comparative de l'énergie entre le calcul stochastique et logarithmique est présentée (Tableau 1). L'étude indique qu'une mise à l'échelle de la conception vers des nœuds technologiques plus avancés pourrait encore réduire la consommation d'énergie, soulignant le potentiel du calcul logarithmique pour des systèmes d'apprentissage automatique écoénergétiques.

Le chapitre se conclut sur une note inspirante, montrant le potentiel prometteur du calcul logarithmique dans la création de systèmes d'apprentissage automatique non seulement écoénergétiques, mais aussi plus précis. Motivés par les résultats obtenus, l'objectif futur est de développer un système plus grand et plus robuste, capable de gérer des tâches réelles sur la puce, nous rapprochant ainsi de la réalisation de cet objectif ambitieux.

<b>Architecture</b>	<b>CLKs Inf.</b>	<b>Précision (%)</b>	<b>E (nJ) Inf.</b>	<b>E (nJ) Inf. &amp; Rd</b>
<b>Stoch Conv.</b>	255	90.0	2.17	2.47
<b>Stoch Conv.</b>	50	86.7	0.43	0.73
<b>Stoch Conv.</b>	25	82.9	0.21	0.51
<b>Stoch PC</b>	255	86.9	0.10	0.40
<b>Stoch PC</b>	50	84.4	0.06	0.36
<b>Stoch PC</b>	20	80.2	0.04	0.34
<b>Logarithmique</b>	1	90.6	0.20	0.50

Table 1: Comparaison des deux machines bayésiennes sur la tâche de reconnaissance de gestes. Conv : calcul stochastique conventionnel. PC : calcul stochastique conscient de la puissance. Inf : inférence.

#### **Résumé du Chapitre 4: Une Plateforme de Prototypage Multimode à Base de Memristors**

Le chapitre 4 explore le domaine innovant et stimulant de la technologie des memristors. Bien que prometteurs pour faciliter l'émergence de nouveaux paradigmes de calcul tels que le calcul analogique, neuromorphique, stochastique, et calcule Proch ou dans le memoire, les memristors se trouvent encore en phase exploratoire, avec des défis intrinsèques et des imperfections. Ce contexte souligne la nécessité de plateformes expérimentales pour le prototypage efficace, les tests et la validation de nouveaux concepts basés sur les memristors.

Le chapitre aborde ces défis de front, en commençant par une exploration perspicace des imperfections des memristors et de leur impact sur la mise en œuvre de nouveaux paradigmes de calcul. Il dévoile des stratégies potentielles pour exploiter ces memristors imparfaits, mettant ainsi en évidence le rôle indispensable des plateformes expérimentales basées sur les memristors (Fig 7). Suite à cette exploration théorique, le chapitre se focalise sur les applications pratiques. Il présente un circuit intégré, conçu comme une plateforme de prototypage à double mode pour les projets de memristors (Fig 8). Cette plateforme intègre un circuit périphérique pour l'intégration des memristors dans les systèmes de calcule numériques et un mode analogique pour une interaction directe avec les memristors (Fig 7). Ainsi, elle offre une base pour le développement et la validation empirique de nouveaux concepts neuromorphiques basés sur les memristors.

Poursuivant le voyage de la théorie à la pratique, le chapitre illustre la conception, la fabrication, les tests



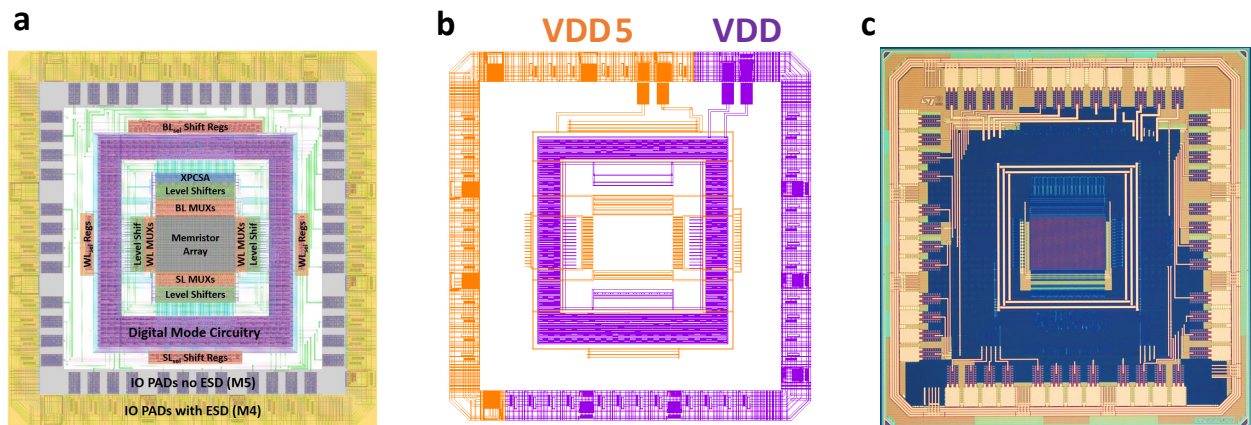


Figure 8: **Plateforme de Prototypage Hybride Multimode Memristor-CMOS Fabriquée.** **a** Vue en plan, **b** Connexions des tensions d'alimentation et **c** Photographie par microscopie optique.

## Conclusion et Projets Futurs

Cette recherche offre un examen rigoureux des problèmes de consommation d'énergie et de fiabilité en IA, mettant en évidence la nécessité de stratégies minutieuses et réfléchies pour l'intégration de l'IA dans la vie de tous les jours. L'étude s'est focalisée sur le développement de circuits intégrés spécialisés pour soutenir les modèles d'IA économes en énergie, en particulier pour les applications de bord.

Un élément central de cette stratégie était l'incorporation de l'inférence bayésienne, une technique d'IA reconnue pour sa transparence et sa responsabilité, renforçant ainsi la confiance dans les applications d'IA. S'inspirant de l'efficacité énergétique exceptionnelle du cerveau humain, nous avons utilisé une architecture de calcul proche de la mémoire, rendue possible par la technologie nanoelectronique avancée. Cette approche a intégré divers domaines, dont l'IA, l'architecture des systèmes de calcul et les technologies émergentes, en exploitant principalement la non-volatilité et les capacités de mémoire proche des memristors.

La contribution de cette recherche à de nouveaux paradigmes de calcul, notamment ceux utilisant des technologies émergentes, est significative. En créant des liens entre les matériaux et les dispositifs, les algorithmes et les systèmes de calcul, elle fournit une voie concrète pour intégrer des dispositifs tels que les memristors dans des systèmes économes en énergie. Elle ouvre la voie à de futures avancées en matière d'efficacité énergétique et de fiabilité de l'IA.

Orientations pour la recherche future :

Plusieurs pistes prometteuses se dessinent pour la recherche future :

Optimisation des systèmes basés sur les memristors : Il est intéressant d'explorer davantage les tests et l'amélioration de la machine bayésienne à grande échelle conçue dans cette étude, avec une attention particulière portée aux nœuds de technologie avancée pour l'efficacité énergétique.

Intégration de nanodispositifs et de la nanophysique dans les systèmes de calcul : Il y a place pour une intégration supplémentaire de nanodispositifs émergents dans notre architecture de machine bayésienne et d'autres modèles d'IA, en commençant par les prototypes SMTJ.

Surmonter les imperfections des memristors : Les imperfections des dispositifs persistent comme un défi important, et la plateforme de prototypage multimode basée sur les memristors développée dans cette recherche offre un outil unique pour relever ce défi.

Aborder les défis de l'apprentissage sur puce : Le circuit intégré développé ouvre une voie pour l'examen de nouveaux concepts et algorithmes neuromorphiques, en exploitant la machine d'échantillonnage MCMC et les réseaux hybrides Memristor/FeFET.

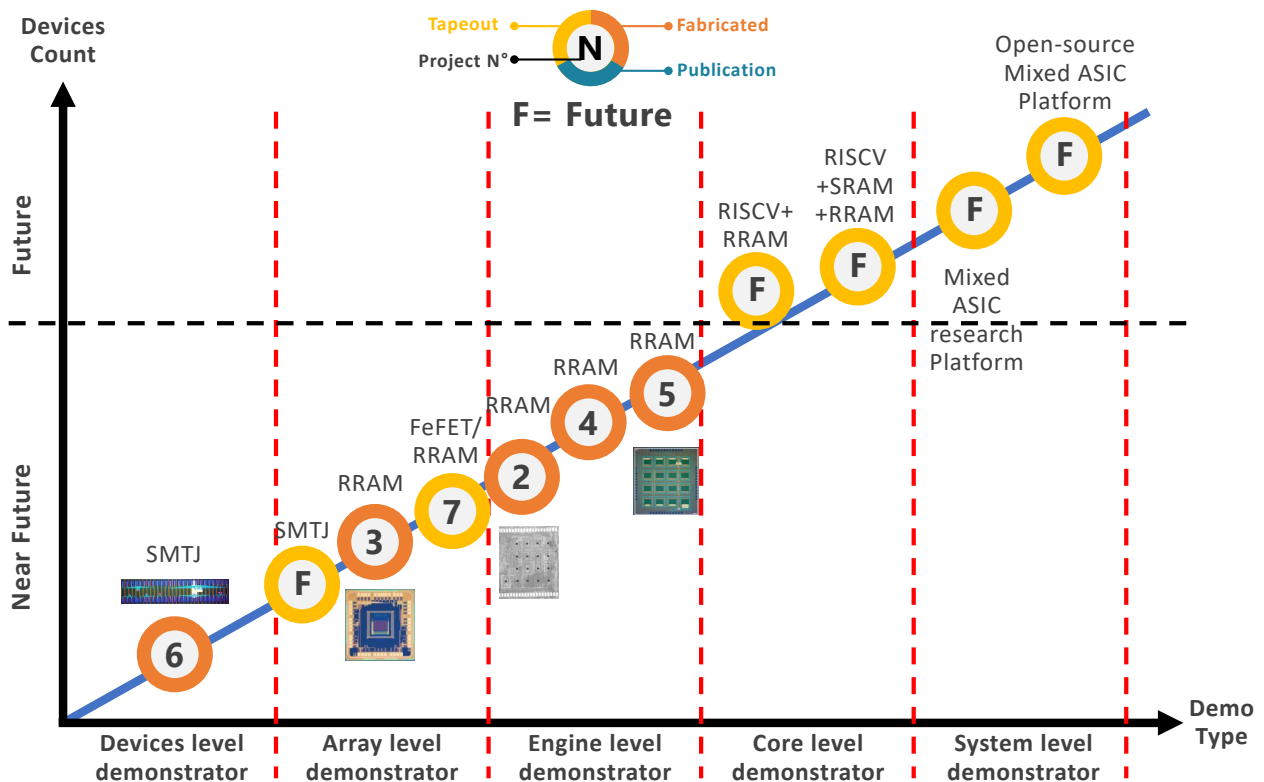


Figure 9: **Un graphique des perspectives** Un graphique prospectif illustrant la croissance projetée de nos efforts de recherche, l'axe des X représentant l'échelle des démonstrateurs du dispositif au circuit complet, et l'axe des Y indiquant le nombre de dispositifs dans les circuits. La ligne linéaire symbolise l'évolution de nos conceptions, des projets existants aux projets futurs, démontrant une augmentation anticipée à la fois de l'échelle et de la complexité.