eFPGAs: Architectural Explorations, System Integration & a Visionary Industrial Survey of Programmable Technologies

Syed Zahid Ahmed

Université Montpellier 2 (UM2)

École Doctorale I2S

LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier)

Domain: Microelectronics

PhD thesis report for partial fulfillment of requirements of Doctorate degree of UM2

Thesis conducted in French Industrial PhD (CIFRE) framework between:

Menta & LIRMM lab (Dec.2007 – Feb. 2011) in Montpellier, FRANCE

"eFPGAs: Architectural Explorations, System Integration & a Visionary Industrial Survey of Programmable Technologies"

eFPGAs: Explorations architecturales, integration système, et une enquête visionnaire industriel des technologies programmable

by

Syed Zahid AHMED

Presented and defended publically on: 22 June 2011

Jury:

Mr. Guy GOGNIAT	Prof. at STICC/UBS (Lorient, FRANCE)	**President**
Mr. Habib MEHREZ	Prof. at LIP6/UPMC (Paris, FRANCE)	**Reviewer**
Mr. Jürgen BECKER	Prof. & CHEO at KIT (Karlsruhe, GERMANY)	**Reviewer**
Mr. Michael HÜBNER	Senior Scientist at KIT (Karlsruhe, GERMANY)	**Examiner**
Mr. Laurent ROUGÉ	Founder & CEO Menta (Montpellier, FRANCE)	**Examiner/Co-Advisor**
Mr. Gilles SASSATELLI	Research Director at LIRMM/CNRS (Montpellier, FRANCE)	**Co-Advisor**
Mr. Lionel TORRES	Prof. at LIRMM/UM2 (Montpellier, FRANCE)	**Advisor**

Short Abstracts in French & English as per requirements of UM2.

RÉSUMÉ

La thèse s'articule autour du thème des FPGA embarqués (eFPGAs). Ce manuscrit analyse les solutions existantes actuellement et discute les challenges et opportunités de ces technologies; une analyse en profondeur des échecs des tentatives passées est également donnée. Sur la base des solutions existantes dans la littérature, une structure de eFPGA à topologie de type grille est proposée, décrite en langage VHDL RTL. Cette solution comporte également les outils de programmation associés. Sur la base de cette proposition, des explorations sont menées quant à la pertinence des solutions proposées au sens de métriques d'actualité tells que densité logique, performance et consommation. Une des contributions notables de cette thèse repose sur la proposition d'une architecture de switch unifiée éliminant les blocs de connexions ainsi que l'interconnexion locale typique des FPGA actuels (telles que ceux modélisables dans le logiciel VPR) tout en autorisant une bonne routabilité. Toutes les expérimentations ont été menées sur une technologie CMOS 65nm faible puissance du fondeur STMicroelectronics, qui permet de fait d'obtenir des évaluations pertinentes. Une seconde contribution notable repose sur l'exploration de l'intégration de eFPGA dans un contexte système sur puce (SoC). Cette approche repose sur l'adjonction d'un eFPGA au sein d'un système intégré, au côté d'un processeur de type LEON3, la programmation s'effectuant sur la base d'une approche de type ESL. Deux explorations sont ainsi déclinées, comme unité intégrée au sein du processeur et comme coprocesseur. Les résultats présentés permettent ainsi d'analyser sous plusieurs angles les compromis possibles ainsi que les perspectives et limitations de ce type d'approches. Finalement, un cas d'étude est également présenté quant à l'intégration de mémoires de type magnétique (MRAM) au sein-même de l'architecture du eFPGA.

Mots-clés: architecture eFPGA, outils CAO pour eFPGAs, enquête industrielle, accélération reconfigurable, ESL, MRAM

ABSTRACT

The thesis extensively revolves around embedded FPGAs (eFPGAs). It conducts detailed survey focused on programmable technologies to investigate potentials and challenges of eFPGAs and probable failure reasons of several past attempts of different kinds. Based on the survey knowledge, technology independent soft eFPGAs of FPGA-like mesh-based classical architecture with standard RTL programming flow are investigated. Detailed eFPGA architectural explorations (including CAD tools) are conducted to explore silicon-efficient (logic density, power, performance etc.)eFPGA architectures. Among notable innovations achieved is unified switch block with complete removal of connection block and local interconnect of classical mesh-based FPGAs (VPR-like) while maintaining good routing efficiency. All experiments are conducted on 65nm CMOS low powerSTMicroelectronics process to get practical silicon values and perspectives. Finally eFPGAs in systems (SoCs) potentials and challenges are addressed. A reconfigurable acceleration scenario with ESL exploitation (for programming ease) and full silicon tradeoffs visualization is presented with integration of eFPGA with LEON3 processor (as a functional and co-processor unit, with also highlighting potential flaws of functional unit in industrial perspectives). An interesting case study for perspectives of emerging MRAM memories for eFPGAs is also presented.

Keywords: eFPGA architectures, CAD tools for eFPGAs, industrial survey, reconfigurable acceleration, ESL, MRAMs

ABSTRACT

(Extended)

Rising design complexities and high manufacturing costs of System on Chip (SoCs) in deep submicron nodes (beyond 90nm) have reached levels where dedicated SoCs can no longer be designed for every application. They must have some post manufacturing flexibility to amortize the high development costs to several end markets. The Field Programmable Gate Arrays (FPGAs) are well known for their flexibility and ease of design modification. With the continuous architectural innovations and Moore's law they have become programmable platforms and in many cases provide a good alternative to implement SoCs directly on them. Unfortunately FPGAs suffer from large silicon gap compared to ASICs/ASSPs. This restricts their use in many high volume applications, and despite enormous benefits of flexibility FPGAs still represent a small niche in industry from revenues stand point compared to ASICs/ASSPs. An obvious choice that comes in mind in such scenario is embedded FPGAs (eFPGAs) to bring benefits of FPGAs right inside SoCs, bridging their challenges for flexibility, product differentiation, time to market etc. However concept of eFPGAs is not new to industry and is historically well known for never succeeding, despite undeniable benefits and potentials.

The thesis work extensively revolves around eFPGAs in three major themes. (i) Survey and analysis of programmable technologies to investigate scope, potential, challenges of eFPGAs. (ii) eFPGAs architectural explorations and tools infrastructure to create efficient customized eFPGAs. (iii) eFPGAs in SoCs investigations and perspectives (including beyond classical technologies/techniques). The general flow and aspects of discussions are as follows.

The thesis presents detailed investigations of FPGAs and eFPGAs research challenges by analyzing state of the art in industrial and academic research. It conducts detailed survey focused on programmable technologies to address potentials and challenges of eFPGAs and probable failure reasons of several past attempts of programmable solutions of different kinds. Based on the knowledge gained from detailed investigations, motivations and potentials of FPGA-like technology independent soft eFPGAs are presented. The CAD tools and graphical infrastructure to create and explore customized eFPGAs with standard RTL programming flow is presented. Detailed eFPGA architectural explorations are conducted to create efficient soft eFPGA architectures with attractive silicon properties (logic density, power, performance etc.). Among notable innovations achieved is complete removal of connection block and local interconnect of classical mesh-based FPGAs (VPR-like) while maintaining good routing efficiency by investigating unified switch block architecture with specialized diverse connectivity of logic block to unidirectional routing network. All experiments are conducted on CMOS 65nm low power STMicroelectronics (ST) process to get practical silicon values and perspectives. Finally eFPGAs in Systems (SoCs) potentials and challenges are addressed in detail. A case study of reconfigurable acceleration is presented with integration of eFPGA with LEON3 processor (as a functional unit and co-processor, with also highlighting flaws of functional unit from commercial stand point). Use of ESL is investigated for programming ease of eFPGA while maintaining standard RTL flow and complete silicon tradeoffs of the conducted experiments are investigated. To further enhance the potentials of eFPGAs for systems, investigations and perspectives of beyond classics emerging Magnetoresistive Random Access Memories (MRAMs) is also briefly discussed as an interesting case study.

Keywords: eFPGAs, FPGAs, CAD tools for eFPGAs, silicon analysis, survey of programmable technologies, FPGAs patents survey, SoC, reconfigurable computing, eFPGAs as co-processors, ESL, MRAMs for FPGAs

Acknowledgements

Finishing a PhD is a great moment of heterogeneous excitements. I thank God for helping me achieve this (my precious!) milestone in such serendipitous and adventurous ways that I have no doubt that treasure is indeed the journey itself. Acknowledgements are the easiest and hardest part of the thesis to write, one can easily write the whole acknowledgements section for almost all theses in just one simple sentence "I would like to thank my parents/family and above all the usual suspects (those of skill will understand anyways)". But I prefer going the other way to well address and exploit the market share of acknowledgements for theses, so I begin my quest.

***Parents*:** First and foremost my parents for their endless love. It's been more than 7 years since I embarked on MS+PhD wonders to foreign lands (reminds me of 7 years of that infamous wizard boy of novels/movies, who ironically also finishes his quest on big screen in July 2011). Sometimes I visited my parents after up to two years of delay due to schedules/visa issues etc. They are my greatest asset. They will hardly understand my thesis at all, but indeed will have the highest joy knowing I did it, whatever it was!

***Advisors*:** I like to thank my mentor *Dr. Gilles Sassatelli*: the man who brought me to France from Germany, perhaps the primary reason I did my PhD from LIRMM, donated me to Menta/Industry. Indeed he deserves the highest market share of credit for this thesis. My acquaintance with him stretches back to late 2004 when I attended his visiting block-lecture in TU-Darmstadt during my masters; it was there I got severely infected with virus of reconfigurable technologies in all forms (later was partly vaccinated by the industry) and was also the first time I programmed an FPGA. I would like to extend my deep regards to *Prof. Lionel Torres*: the supreme commander of this thesis, a born leader with multitude of research interests and strong eye on industry too. During a significant part of my thesis he was also serving as director of microelectronics department of LIRMM and I used to be fascinated observing his exceptional email skills to steer things. His continuous influence and advice to do stuff like: present this whole concept in 10 slides; explain this entire scenario in few lines etc. helped me practice such skills that were invaluable in making fine publications and also this thesis report. I will also like to deeply thank both Gilles and Lionel for: their everlasting patience for/to my industry vs academia hysteria (due to industrial PhD), my life time absence award presence in the lab, reviewing of my research papers to correct/transform their industrial sauce language to a form digestible by peer scientific reviewers (and they were, we almost never lost!). They also provided me opportunity to occasionally take part in conferences program committees that further helped me understand back-end of conferences, review process and also psychology of reviewers.

***Menta*:** I would like to express my gratitude to *Mr. Laurent Rougé*, primarily for founding Menta or none of this would have happened! (see prologue). My stay at Menta was full of heterogeneous experiences, challenges and opportunities making my industrial PhD a true industrial PhD, helping me get trained as a student of industry (see epilogue). Being almost of my age Laurent was always more a close friend than boss. Other than our countless hours of technical/business discussions, he shared with me his enormous general knowledge and also provided me substantial knowledge about France. My research work would never had been possible without the help of my colleagues (and also several master thesis students that came along the way) at Menta (co-authors of my papers). I will always remember the wonderful time spent with Jean-Baptise Cuelle, Julien Eydoux and Alexandre Martheley, particularly our quest of eFPGA Creator. Finally I thank Menta as a whole; it was a pleasure to see/have my thesis a foundation part of it and hope it will help partly guide future strategies.

***Jury*:** I would like to express deep regards to my distinguished jury members. Firstly thanks to *Prof. Guy Gogniat* for accepting being president of the jury, giving memorable final remarks on the thesis work on behalf of jury and announcing me Dr. I would like to thank *Prof. Habib Mehrez* for accepting to be reviewer of my thesis. Being a well renowned authority and one of the very few professors in France/Europe deeply involved in FPGA architecture research domain his presence in jury was like a LUT in FPGA. I am thankful for his wonderful remarks on architectural explorations but perhaps will especially cherish forever his appreciating remark on my relatively unusual chapter 2 in the light of CIFRE PhDs [5.11]. It was a privilege to get such remarks from a very senior professor like him with decades of experience. I will like to express my heartiest regards to *Prof. Jürgen Becker*: a world renowned authority on industrial-driven FPGA-based reconfigurable systems, premier conferences chair etc. for accepting to be the 2nd reviewer of the thesis on a very short notice and despite his enormously busy schedule due to his additional admin responsibilities. I will always remember his enchanted mail from Washington DC to Frankfurt flight in which he sent confirmation to jury members that he will be delighted to serve as reviewer. I am thankful for his wonderful remarks on thesis report and appreciations of this thesis's understanding of industrial scenarios and challenges. I like to express my gratitude to *Dr. Michael Hübner* (the infamous apprentice & colleague of J. Becker): an iconic expert of Xilinx FPGAs, reconfigurable systems, peer conferences chair etc. to accept serving as examiner in the jury. Together Jürgen and Michael in addition to making my jury international brought Germany in it, the country where I fell in love with microelectronics and programmable technologies. Finally I will like to express my regards to the jury for their admiration of the multidisciplinary contributions and efforts of this thesis.

***Academia*:** Perhaps the most interesting part of PhD training is conferences, providing chance to meet leading experts from around the world. It is indeed hard to go in the long list but I like to mention few with whom I had quite interesting discussions on FPGAs/broad spectrum of programmable technologies/academia etc. which were very helpful in my research. In alphabetical order: A. Agarwal (MIT), K. Benkrid (U. Edinburgh), E. Cota (UFRGS), O. Hammami (ENSTA), L. Lagadec (UBO), G. Lemieux (UBC), R. Leupers (RWTH Aachen), F. Moraes (PUCRS), F. Morgan (NUIG), M. Lubaszewski (UFRGS), K. Torki (CMP). I like to thank Prof. R. Tessier (UMASS); detailed informal discussions on his summer 2009 visit to LIRMM were highly interesting. He gave several details about FPGA research, USA academia, general tips about PhDs and publications & my infinite questions about J. Rose. I will also like to express gratitude to Prof. M. Robert (LIRMM), master of my masters! for his general tips (particularly his remark "too many slides kill slides") on making nice presentations and art of extra slides. I will also like to re-thank Prof. M. Glesner (TUD); I could not have done this PhD without the foundation knowledge of microelectronics that I learned in his institute during my Masters. Finally I like to express my deepest regards to the two great professors who mesmerized me the most (technically and their visionary unconventional styles) in my PhD. Unfortunately I only electronically met them: Prof. *Jonathan Rose* (U. Toronto) & Prof. *Reiner Hartenstein* (U. Kaiserslautern). I can write an entire chapter on them but have limited silicon budget for acknowledgements and indeed *cannot exploit Moore's law on font size for obvious reasons!* In short: immense thanks to J. Rose for his vision of sharing his publications online with the community & the infamous VPR book. Countless thanks to R. Hartenstein for his amazing keynotes (drove me crazy), contributions to education, *Linkedin* relationship and not killing me for stealing "visionary" from his keynotes and put it in my thesis title. I dedicate my FPL10 survey presentation [S-3b] (slides were fortunately published! as part of FPL's 20th birthday initiative) to them both in general and in particular slide 26 (education crisis) to R. Hartenstein.

***Industry*:** The story of my PhD is incomplete without the hypothetical Director-X (Semiconductor Industry). Looking back now to my PhD in some ways I have to admit that my biggest challenges and troubles came from the industry and were also partly solved by the industry (and story still goes on I guess). Just on the mere 2nd day of my PhD I had to directly encounter the industry [S-2], it was a strange and sudden shock, things were too different than much softer academia and perhaps like a fairytale story I was instantly in a new world I had no idea of. However in that strange world I came across counterparts of our academian world, on the top of which is UBM (United Business Media) empire (owner of EEtimes and several other EEtimes-like electronic media) and also similar efforts of companies for teaching consumers, physical technology and business experts etc. It took a while before I reluctantly shook hand with this strange master as a student of industry and it took me to a free! magic carpet ride to this strange new world "*so hard to live in, so hard to leave*". I guess now I have partly realized why we call it real-world. I am highly thankful to Director-X for all the trainings and skills, they were priceless to deal challenges of my PhD; I owe in particular my chapter 2.3 to him. Coming from hypothetical to real (the musketeers of Director-X), I would like to express my regards to some of the fascinating people from technology/business I came across in scientific/industrial conferences, discussions with them were highly rewarding in my thesis. In alphabetical order: V. Betz (Altera), M. Dickinson (Altera), V. Kathail (Synfora), H. Kim (Samsung), H. Krupnova (ST), P. Kundu (Intel), P. Lysaght (Xilinx), G. Martin (Tensilica), M. Muller (ARM), HN.Nguyen (Bull), I.Phillips (ARM), G. Saucier (D&R), C. Schmitt (eSilicon), J.Tully (Gartner)., H. Wildt (fenix-da). I will also like to express gratitude to the infamous industry bloggers in related domain, to name just few (in alphabetical order): P. Clarke (EEtimes), E. Esteve (SemiWiki), D. McGrath (EEtimes), D. Manners (EltectronicsWeekly), R. Merritt (EEtimes), C. Maxfield (EEtimes), D. Nenni (SemiWiki), R. Wilson (EDN), J. Yoshida (EEtimes). Their wonderful articles taught me a lot. Finally I thank EEtimes as a whole for its open and social structure, webinars, virtual conferences etc. I wish one day there is UBM-like UAM (United Academics Media).

***Friends*:** There is nothing like friendship in the world. In the usual darkness, stress, frustrating moments during thesis it's the loving friendships that come to rescue. It is near impossible for me to go in the details of all the friends in/out of Montpellier, France, Euorpe, World (haven't met any alien yet!) and I am afraid I will hardly be able to do justice in mentioning names so I use a trick partly borrowed from what we do in research papers for author names. My dearest @facebook, @linkedin et al, you mean more to me than dead silicon and there is indeed lot more to life than research.

***ANRT*:** Finally I like to thank ANRT/CIFRE [5.11] for providing me strong motivations and **revenue** for this phenomenal complex adventure to get trained for research and industry, which I frequently used to dub during my PhD as "Industry vs Academics (|V|): the inside story of my PhD (*see epilogue*!)". The ANRT's ROI hopefully will be highly justified in years to come & I wish to be/remain a close friend of academia in industry.

S. Z. Ahmed; 13 July 2011

*“**Ph deDicated to …** Dreams of Grad Students”*

Family:

In loving memory of my late grandfather, Syed M. Sami Ahmed (died in 2002) who always wished of me to become a Dr. of medicine (often termed the real Dr.), when I preferred and took Engineering he wondered if I become *the other Dr*. Now I guess/hope I shall be providing treatments to/for Sick-SoCs.

Table of Contents

List of Figures

List of Tables

Prologue

The origins of this thesis stretch back to the official founding of Menta in July 2007. When this thesis was created between Menta and LIRMM (University of Montpellier) under the umbrella of the CIFRE (in English: Industrial Contracts for training through Research) process of French Ministry of Higher Education and Research run by ANRT [5.11], the work started from December 2007. This created the first contracted employee/scientist for Menta and made the company from one man to the famous industry term "two men in garage". Menta has matured and expanded over the years and at present (calendar Q2 2011) is the only known existing soft-eFPGA provider company.

The motivations and origins of this thesis work were very exciting and challenging. The long-term road map unknown, directions unknown, partners unknown, markets unknown, competitors almost unknown (in terms of existence), Industry soon entered into 2008-2009 global economic crisis. The almost known tagline was to make embedded FPGAs and "*everyone who tried to do this died or left the business*". This thesis is not a Holy Grail for Menta, but indeed has tried to answer several questions and created some solutions. The startup nature of Menta, the economic crisis philosophical influence, the research knowledge of LIRMM, help from hypothetical Director-X (Semiconductor Industry, see acknowledgements section) with articles/blogs/press news/industrial events etc. along with several other factors (intent is providing perspective not autobiography) combined and created an almost ideal CIFRE PhD scenario which has theme motivation "*You will become a doctor once you have defended your thesis, proving that you are capable of undertaking a substantial R&D project. You will be recognized for your participation in research and innovation, your keen capacity for tackling the problems facing a company, interaction with partners from different cultures. You will learn how to deal with the real-life situations your company faces, and acquire behavioral and interpersonal skills in the process*" [5.11]. This thesis report is the connection of dots looking backwards to the knowledge gained.

***"I"** will be back at END in **epilogue**.*

Chapter 1: Introduction

This chapter presents the global overview, motivations and contributions of this thesis work. Section 1.1 provides the challenges that are currently faced by industry in scenario of this thesis and what observations can be drawn from them for research motivations and objectives. Section 1.2 describes the major contributions of the thesis work and finally section 1.3 provides the outline of the thesis report.

1.1 Motivation & Objectives

Rising design complexities and high manufacturing costs of System on Chip (SoCs) have reached levels where dedicated SoCs can no longer be designed for every application. They must have some post manufacturing flexibility to amortize the high development costs to several end markets [S-1]. The Field Programmable Gate Arrays (FPGAs) are well known for their flexibility and ease of design modification. With the continuous architectural innovations and Moore's law they have become programmable platforms and in many cases provide a good alternative to implement SoCs directly on them. However FPGAs suffer from large silicon gap in terms of Area, Power and Speed [Kuon&Rose09, 5.2] compared to ASICs (Application Specific Integrated Circuits), ASSPs (Application Specific Standard Products). This restricts their use in many high volume applications despite enormous benefits of flexibility. Although FPGAs being a steadily growing market, still represents just about only 2% of the semiconductor industry and as of 2009 represents around 4 billion dollars market compared to more than 80 billion dollars market of ASICs and ASSPs [2.1][S-3b].

However as stated above, it is becoming more and more essential for SoCs to have some post manufacturing flexibility for addressing key issues like product differentiation, time to market etc. With an added challenge of power consumption, that has become most crucial issue in industry and forcing the SoC designs to be more and more heterogeneous and customized to be optimal in power budgets. This gives dual challenge to SoC designers. On one side the flexibility is required to meet tough market challenges and on the other side solutions should be as custom and non-flexible as possible to meet silicon budgets.

An obvious choice that comes in mind in such scenario is embedded FPGAs (eFPGAs). eFPGAs allow bringing the well known FPGA benefits right inside the SoCs. However immediate question regarding eFPGAs is, how sound that idea is from technological and commercial perspective as the concept of eFPGAs is not new and well known in industry for never succeeding. What are the challenges and reasons that might have been the reasons of failures in the past and what can be learned from them as inspiration to avoid doing same mistakes. The goal and challenge for the SoCs is to seek some flexibility by having some programmable portions or IPs. FPGAs are most famous and dominant symbol of programmable devices but are not the only programmable devices. These challenges and observations lead to different questions and investigation motivations for eFPGA research, for instance.

- What is the semiconductor industry landscape and how different technologies fit in it?
- What are different types of competing programmable logic solutions? How they compare, what is their status and what are ongoing trends in industry and academic research?
- Are FPGAs dominant in programmable logic space, if yes what makes them dominant?
- Several efforts and innovative solutions in terms of programmable logic and reconfigurable computing have failed in past, what could be the reason and what can be learned from that?
- Should eFPGAs follow FPGA-like conventional architecture, if yes what are the pros and cons?
- How to create eFPGAs effectively? What lessons and motivations can be learned from the past?
- How sound eFPGAs concept is for SoCs? What are the challenges, opportunities and perspectives?
- Can some beyond classics emerging works/technologies help to solve/improve some classical problems related to FPGAs/eFPGAs? What can be the challenges and perspectives?

Based on such diverse rough motivations and challenges the thesis contributed in several topics, which finally led to three main axes of contribution that are discussed below.

1.2 Contributions

The contributions of thesis are graphically illustrated in figure 1. The thesis extensively revolves around eFPGAs, to investigate and address the objectives of the thesis; the research done can be divided into three major blocks or axes. They are represented as *Red [R], Blue [B] and Green [G]* in the figure. Brief details of contribution areas of figure 1 are explained below.

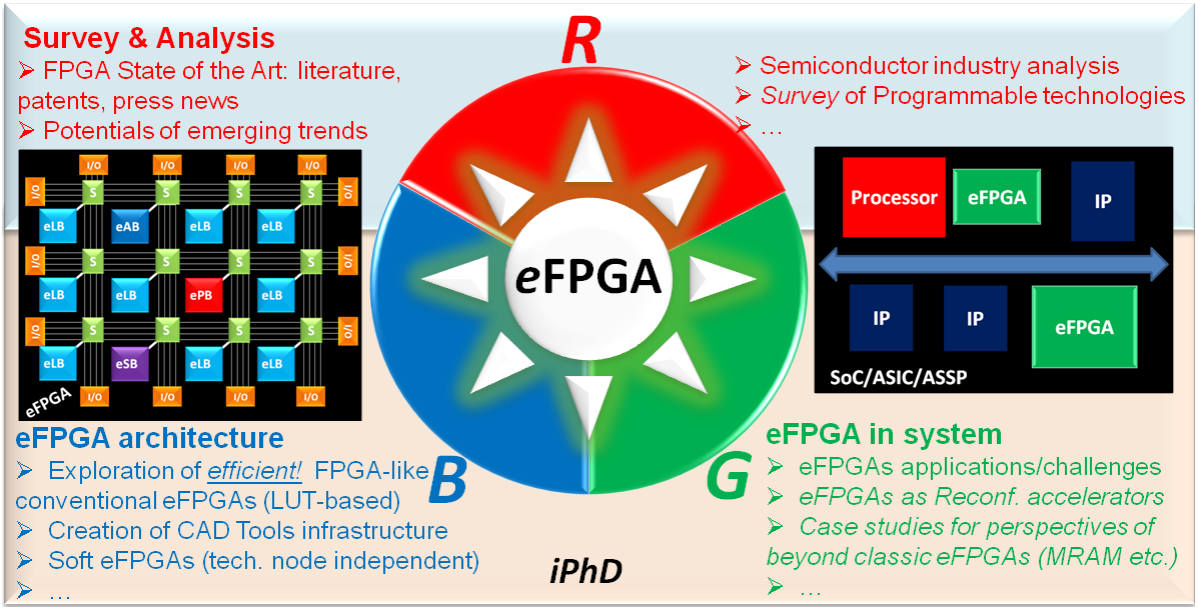


Fig. 1: Research Blocks Graph of thesis contributions

Survey & Analysis [R]

The R axis acts as an umbrella covering the motivations and research directions, helping make well informed research decisions in the light of survey knowledge. It fundamentally addresses following aspects.

- Study of state of art FPGAs in detail (literature, patents etc.) to find research challenges
- Study of changing technological trends and challenges in industry (state of state of art) to get informed and trained for real-life challenges
- Analyze potentials of different programmable technologies, inspect how/where eFPGAs fit in the spectrum and investigate the causes of failures of several past solutions in research and industry

This axis partly covers the obligatory part of almost all theses (analyzing basics of the state of the art in that area), with additional emphasis on survey of industrial solutions to well categorize and understand real-life challenges of/for eFPGAs and other programmable technologies in a broad spectrum.

Key Contributions:

- ❖ In depth study of literature of leading FPGA vendors (research/commercial articles, patents etc.). Detailed study of academic research on FPGAs
- ❖ Investigation of eFPGA/eFPGA-like efforts, solutions from industry and academics, visualizing probable reasons of past failures
- ❖ Comprehensive survey of semiconductor industry, focused on programmable technologies with investigation of strengths and weaknesses (including failures of past) of different solutions
- ❖ In context/learning of/from survey research, specialized design of references section of thesis

eFPGA Architectural Explorations [B]

The B axis extensively deals for the research of eFPGA architecture to create efficient customized eFPGAs, it is the biggest contribution axis of this thesis work in terms of efforts done and time spent. It fundamentally addresses.

- Architectural explorations and innovations using FPGA CAD and silicon investigations to create efficient FPGA-like (LUT-based) technology independent soft eFPGAs
- Focus on beyond CMOS 90nm nodes (65nm was mostly used) to visualize static power challenges
- Compare and tune architectures to be more area, power, timing efficient for benchmarks applications
- Create tools infrastructure to facilitate and accelerate research investigations

This axis is similar to classical FPGA architectural research. It deals with architectural innovations with the CAD and silicon visualizations of the architectures on advanced nodes (beyond 90nm for judging issues of leakage etc.) of *real* silicon process by implementing the architecture and analyzing the silicon properties of those architectures in terms of area, power and performance. Numerous tools were created to facilitate the exploration research. This axis performs combined CAD and silicon investigation for understanding the pros and cons of the under investigation architectures in terms of architectural efficiency (mapping benchmark applications) and silicon efficiency (logic density, speed, power). Qualifying both is essential; architecture must succeed implementing benchmarks to qualify for the chosen architectural parameters (LUT size, cluster size, channel size, routing architecture etc.) to be appropriate for the target domain for which eFPGA is being created, and on the other hand the proposed architecture must have attractive silicon properties in terms of area, power and performance to be value added proposition for the end product (SoC, ASSP, ASIC etc).

Key Contributions:

- Exploration of unified SB (switch block) based architecture with eliminated local interconnect (LI) of logic block which is found in CB (connection block) based VPR-like architectures [4.1]
- Soft technology independent architecture (multiplexer based routing, latch/flip-flop based configuration) with unidirectional/single-driver routing architecture
- eFPGA Creator™: tools suite to create and explore customized eFPGAs. This thesis contributed
  - Motivations for advanced GUI based tools infrastructure for exploration
  - Full/partial design of several portions of the tools suite for custom architectures exploration
  - Automatic hardware generation (VHDL and scripts) of custom architectures
- Detailed analysis of fundamental architectural parameters (LUT size, cluster size, channel size etc.) on STMicroelectronics CMOS 65nmLP process for benchmark applications (MCNC [4.1]) for general overview
- Exploration of connection of logic block to the routing architecture (through unified SB) in depopulated diverse ways to highly enhance silicon efficiency (rivals past published results)
- Exploring power vs speed issues and challenges with change of threshold voltage (LVT, SVT, HVT) and process node type (LP, GP) on STMicroelectronics CMOS 65nm

eFPGA in Systems [G]

Since the thesis work is focused on embedded FPGAs, the G axis investigates the challenges and opportunities (from knowledge of R axis) of eFPGAs in systems scenario. It fundamentally addresses.

- eFPGAs in SoC scenario as an IP
- eFPGAs as reconfigurable accelerator
- System integration challenges (in terms of physical integration, and value addition vs silicon tradeoff)
- Investigation of perspectives of beyond classics technologies/methods for enhancing eFPGAs capabilities and potentials

The work and knowledge of B and G axes are highly interlinked. B quests for creating efficient eFPGAs and G visualizes how efficient they will be in some real scenarios. The philosophy behind the eFPGA of Menta is to create soft target independent customized eFPGAs programmable by standard flows for everybody and suitable for multiple potentials as seen in the concept system diagram with eFPGAs in figure 1. This axis investigates the potentials and challenges related to that in general and in detail the potentials of reconfigurable acceleration. It also highlights and investigates the perspectives of beyond classics emerging technologies to help solve some conventional problems and enhance eFPGAs potentials.

Key Contributions:

- General discussions for potentials of eFPGAs for industry, reconfigurable computing (reconfigurable acceleration with standard RTL programming flow with ESL exploitation)
- eFPGA interfacing with processors (as functional unit and co-processor unit), perform experiments (HW/SW co-design) with full silicon tradeoffs visualization on ST 65nmLP
- Highlighting in industrial scenario: ESL benefits, potential flaws of reconfigurable functional unit, system integration challenges and issues for eFPGAs
- A real and innovative case study for the potentials of beyond classics emerging MRAMs for enhancing eFPGAs capabilities (non-volatility, dynamic reconfiguration, multi-context, fabrication ease etc.), an interesting compliment for the eFPGA research work of this thesis

1.3 Thesis Organization

The thesis is organized in a suitable way to address the ideas, contributions and easier for the reader to understand. In addition the chapters are clusters of mini chapters to further elaborate the exact distinct objective of the chapter based on figure 1, which leads to three distinct chapters (with CAD infrastructure presented as separate chapter for ease of discussions). Furthermore all sections and subsections of the chapters in most cases are further characterized to effectively present the center ideas immediately for the reader. The outline of the chapters is as follows.

Chapter2 details the contribution ***R*** of the thesis work. The chapter itself is composed of the three fundamental axes of the thesis [RBG] divided in three core sections. Section 2.1 outlines research challenges of FPGAs, as eFPGA has a classical LUT based architecture so understanding the FPGA challenges is obligatory. This section covers these aspects in detail in both industrial and academic scenario. Section 2.2 addresses the issues and work done with embedded FPGAs, which is essential for this thesis as it deals with embedded FPGAs so additional challenges regarding eFPGAs must be understood. Finally section 2.3 details the industrial survey and will address the major findings of the commercial research. It presents general overview of industry and its markets; survey focused on programmable technologies highlighting their strengths and weaknesses and finally investigates how/where eFPGAs fit in the spectrum of current changing trends in the industry.

Chapter3 is an overview chapter of the CAD infrastructure of eFPGAs (partly and closely related to axis B contribution). It provides the fundamentals of eFPGA architecture, its programming flow and tools which program it and then in detail the eFPGA Creator with its user friendly graphical tools suite which helps to create customized eFPGA architectures. It will provide a comprehensive overview of the motivations, differentiations of the tool and its capabilities to create, analyze and implement customized eFPGAs, and contributions of this thesis work in that regard.

Chapter4 addresses the research contribution ***B*** in detail which is the largest contribution of this thesis work in terms of time spent and experiments conducted. It describes the detailed experiments conducted for eFPGA architectural exploration in the light of knowledge gained from chapter 2 and with the help of eFPGA Creator (chapter 3). Section 4.1 presents general experiments to understand and investigate several

fundamental research challenges with FPGAs. Later sections detail architectural research done using eFPGA Creator for creating complex customized architectures which have higher density, higher speed and lower power compared to basic customized architectures of section 4.1 while still being capable of efficiently routing benchmark applications. All experiments are fully analyzed on ST65nm low power process for area, power and speed tradeoffs.

Chapter5 presents the research contribution ***G***. Section 5.1 provides general motivations and potentials of eFPGAs for SoCs, in particular as a reconfigurable accelerator. Section 5.2 presents detailed experiments conducted with eFPGAs and LEON3 integration. Section 5.3 outlines the system integration challenges in the light of industrial survey and experiments knowledge of the thesis. Finally the chapter presents the case study of using MRAMs (Magnetoresistive RAM) based eFPGAs in SoCs. It also briefly outlines the test chips that were taped out based on eFPGA architecture explored by this thesis (130nm MRAM based configuration test chip, 65nm latch/SRAM based configuration test chip).

Chapter6 provides the conclusions and future outlook.

References section is characterized and partitioned, it can also serve as a partial standalone chapter role. The citation of references in text is done with numerical reference (section.ref number) and sometimes additional information is augmented with the numerical reference for added ease of reading in context. The form used in general is [AuthorYear, ref.] or [Company_AuthorYear, ref.]. Self citations have the form [S- number]. This creates a customized domain specific solution, combining benefits of several styles often used in citations.

Disclaimer/Notice

Chapter 2: Survey & Investigation of Objectives [R]

The first step of research is to analyze the state of the art in that area and the contributions done by other researchers. This chapter discusses deeper investigations of the three foundation axes (RBG) of this thesis work explained in chapter 1.

Section 2.1 explains the fundamentals of FPGA architectures and challenges. First it explains the architecture fundamentals and explain them in the light of advances in the state of the art including the post 90nm silicon challenges that are becoming the biggest challenge to all semiconductor industry in general so understanding these challenges is obligatory as wrong research conclusions or directions can easily be made if not effectively considering them. The research of FPGAs is virtually impossible without FPGA CAD tools, so understanding the fundamentals of them is essential. In this regard FPGA CAD research done by highest cited academics is discussed along with architectural challenges addressed by several academic works. Recent trends have shown that to meet the FPGA challenges research has also been conducted in beyond classics technologies, a brief overview of these approaches will be discussed and among them MRAMs and their potentials in bit more detail in this regard, which is one of complimentary contributing part of this thesis work in terms of beyond classics perspectives.

Section 2.2 discusses the embedded FPGA/embedded FPGA like solutions. In this regard very few success stories exist in industry, it will try to identify those approaches with their pros and cons. In principle more efforts are done in this area by academics than industry in different dimensions, it will present brief overview of these efforts in general and more relevant to this thesis work area in particular to find research motivations.

Section 2.3 addresses a differentiating part of this thesis work which serves as the foundation basis of the research directions and motivations. It will briefly discuss the efforts for understanding and integrating in industry, a survey of patents study of FPGA vendors, a comprehensive overview of semiconductor industry markets and a survey of programmable technologies in technical and especially commercial scenario.

2.1 FPGAs State of Art

The section first provides the almost vendor independent basic fundamentals of FPGA architecture and then comprehensively surveys the architectural enhancements and challenges addressed by state of art FPGA vendors and academics research (separately addressed for more clearly identifying their contributions) to find research challenges and motivations. Finally it also addresses an overview of some beyond classics researches underway in industry and academics (like MEMS, NoCs, MRAMs etc.) to investigate their potentials for solving some FPGA challenges. The outline is as follows.

Section 2.1.1 provides the fundamentals of FPGA architecture and terminologies which are not standard (all vendors use their own commercial names) but quite well known and recognized among research community to identify the research target. This section will help to understand later more advanced sections and also the research work conducted in this thesis.

Section 2.1.2 addresses the key advancements of the state of the art FPGAs through the years and latest new challenges in addition to fundamental ones due to growing issues of power consumption in beyond

90nm era. The thesis has considered mostly Xilinx and Altera in discussions as they are the market leaders with more than 80% market share [Coudert09, 2.10] and are most advanced in technology.

Section 2.1.3 will present several well recognized/cited research efforts from academics to address several key challenges of FPGAs architecture, in particular FPGA CAD. This further gives the insight to research challenges in addition to general overview from the state of the art research from section 2.1.2.

Section 2.1.4 will briefly address research efforts in industry and academics to use beyond classic technologies (MEMS, MRAMs, NoC etc.) to address some of the key challenges of FPGAs. We will concentrate bit more on MRAMs which have been used in close partnership in this work to conduct some beyond classics experiments. They will be further addressed in more detail in chapter 5.

2.1.1 FPGAs Fundamentals

Field Programmable Gate Arrays (FPGAs) have greatly evolved since their invention and commercialization in late 1980s ([Xilinx_Freeman89, 3.1]) and have become complex programmable platforms. However the fundamentals have remained similar. This section provides a brief overview of the fundamentals of FPGA architecture in vendor independent manner to highlight the key issues of architecture and terminologies that are widely common among scientific literature. They will help to grasp the main ideas quickly and will help in later discussions of other researches (industrial and academic) and explanations of this thesis work in later chapters. All FPGA vendors have their own commercial names for these fundamental elements and architecture styles that differentiate one from the other.

In a nutshell the foundation of FPGA can be described as a composition of three elements. First a logic block which has the Look-Up Table (LUT) to implement Boolean functions, paired with a Flip-Flop (FF) providing sequential behavior. The logic block is often composed of multiple LUT-FF pairs (logic elements) and usually referred as cluster in scientific literature. Second the programmable interconnect/routing architecture connects these clusters to route data. The third element is programmable I/Os which provide access to outside world.. The programming of FPGAs is mostly done by HDLs like standard ASIC flows of synthesis, place & route (PAR), and final step of bit/configuration generation makes FPGAs different than ASICs (CAD flow will be addressed in detail in section 2.1.3). The details of above mentioned architectural fundamentals along with some additional details are described below with the help of figure 2.1a and 2.1b.

Island Style/Mesh based Architecture

Figure 2.1a shows the generic block diagram of mesh based FPGA architecture [Betz&Rose99, 4.1]. The topology is often referred as “Island Style” architecture in the literature where the logic blocks are surrounded by a sea of routing resources (connection block, switch block and routing channels) as shown in figure. The Island Style topology is the most widely used FPGA topology and is common among all leading FPGA vendors. The other less common topology is Hierarchical/Tree based (will be briefly addressed in section 2.1.3). This thesis work is also based on Island Style architecture. A brief detail of building blocks of the architecture is discussed below.

Logic Block/Cluster

The logic block (LB) is the main computational element of an FPGA that provides the fine-grained reconfigurable flexibility of an FPGA. In principle it is a Look-Up Table (LUT) + Flip-Flop (FF) block that implements combinational/sequential Boolean logic of the mapped application. The LUT + FF pair is frequently referred as a logic element (LE) or a basic logic element (BLE) in the literature. The logic block can be a single BLE, but in practice is often composed of multiple BLEs and is called a cluster in literature and cluster size is an important architectural parameter which describes how many BLEs are in a logic

block (LB). Figure 2.1a depicts the fundamentals of BLEs and LB. In our discussions in later chapters we will frequently use the following terms.

- *LUT Size*: The size of LUT (3, 4, 6 etc.) in the LE/BLE, often referred as K
- *Cluster Size*: Number of LUTs (LEs) inside the cluster, often referred as N

Routing Architecture

Routing in some ways is the most challenging and exciting part of FPGA research. It is also the most expensive part in terms of silicon. The routing architecture can be divided into three fundamental parts. In scientific literature they are often referred as; switch block/box (SB), connection block/box (CB) and the routing channels/tracks. They are depicted in figure 2.1a; a brief overview is presented below.

- ***Switch Block***: The switch block (SB) is the main routing hub that controls the data traffic on routing channels by connecting data on the routing tracks to other tracks
- ***Connection Block***: The connection block (CB) is closely linked with SB from functionality perspective, and that is switching data. The SB connects different tracks with one another for routing the data and CB provides the data access for routing tracks to/from the logic block (LB). CB is often partially and sometimes fully merged with SB as their functionality and purpose is similar in many aspects (will be addressed further in later sections/chapters)
- ***Routing Channels***: The routing channels are the actual physical metal lines that create the data highway of the FPGA in connection with CB and SB. The amount of routing tracks in a routing channel are often referred as channel width/size and often represented as W in scientific literature
- ***HOP***: The routing HOP (long jump) is an added parameter/property of routing architecture that decides/depicts properties for long wires. The fundamental architecture shown in figure 2.1a has a HOP of 1 only as the communication in one jump is only possible between direct neighboring LBs. HOP will be addressed in more detail in later sections

Programmable I/Os

The programmable I/Os provide connectivity to outside world. The I/Os are often connected to the routing architecture through CB in a similar manner like LB. Since the routing channels are immense in size for realistic architecture and in principle can provide thousands of I/Os (physically impossible to manage due to package size), connection like that allows automatically an I/O reduction and control of the quantity of I/Os depending on the target needs.

Routing Driver Styles

The above discussions have just described the fundamentals of FPGAs. There are numerous higher levels and architectural complexity issues in real scenarios (some will be addressed in later sections). It will be interesting to explain one a bit higher-level aspect along with the basic fundamentals in this introductory section to facilitate ease of reading and discussions of later sections and chapters. Figure 2.1a depicts basics of SB and CB which were discussed above. Figure 2.1b provides a slightly more detailed overview of how they are actually implemented and work in connection with the LB. The figure illustrates two types of routing architectures that are referred as bidirectional (multiple drivers) and unidirectional (single driver) in the literature. Historically bidirectional routing remained widely used in architecture research and commercial FPGAs. However modern FPGAs use mostly unidirectional (or a mix of both) as they have been found to be superior compared to Bidirectional (contrary to prior beliefs) both by industrial and academic research, [Altera_Lewis03, 1.26] and [Lemieux04, 4.19] have addressed the issue in detail with experimental proofs. A brief distinction among the two styles can be described with figure 2.1b as.

- In a bidirectional routing architecture, multiple elements can drive a routing channel (hence also called multiple drivers routing). It can be seen from figure that the output of LB is connected to the routing tracks with pass-transistors allowing signal to flow in both directions. The tri-state buffers in conjunction allow a continuation or break point for a track.
- In case of unidirectional routing, each routing track has only one driver (hence also called single driver) and that is a multiplexer which makes data to flow in only one direction on the track. The above mentioned references describe in detail how unidirectional outperforms bidirectional in almost all aspects (area, power, speed). In general since in reality the signal only flows in one direction in channels after an application is mapped (rarely a time multiplexing is done). The enhanced flexibility, added overhead costs of tri-state buffers, large pass-transistors and added configuration needed for bidirectional routing is useless. This thesis work is also focused on unidirectional routing architecture.

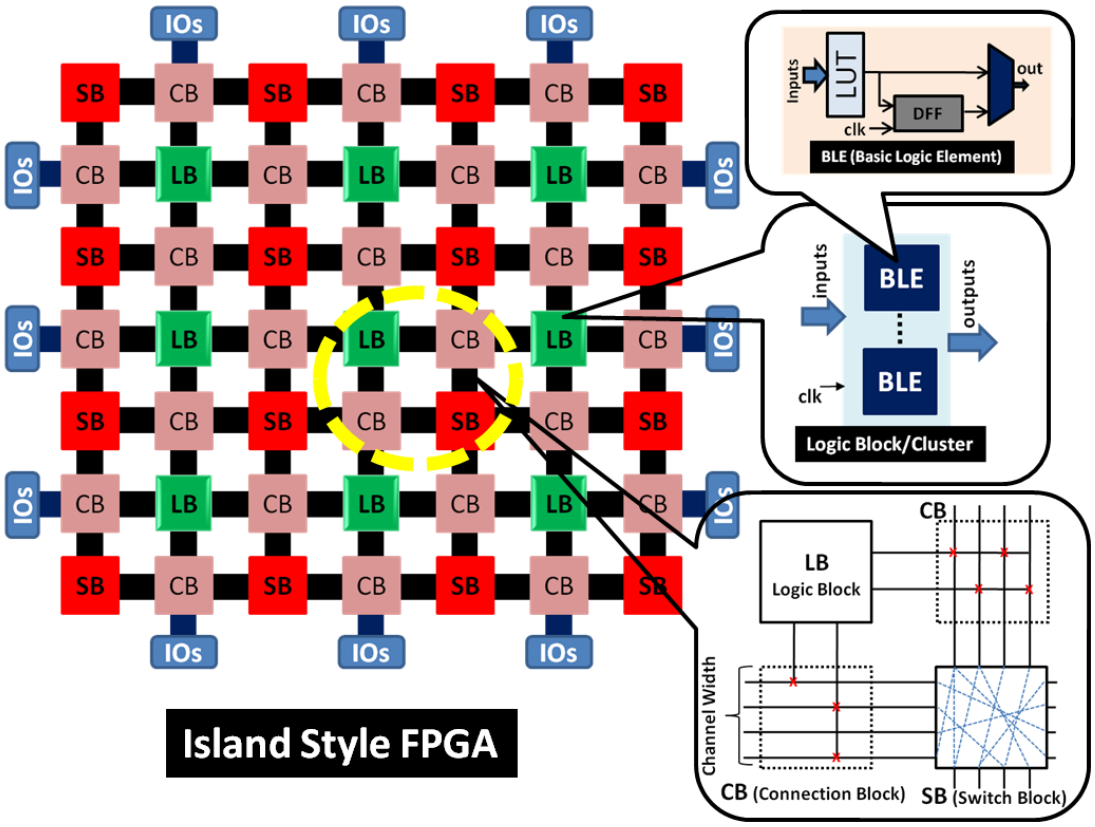


Fig. 2.1a: FPGA architecture fundamentals: Island Style Architecture [Betz&Rose99, 4.1]

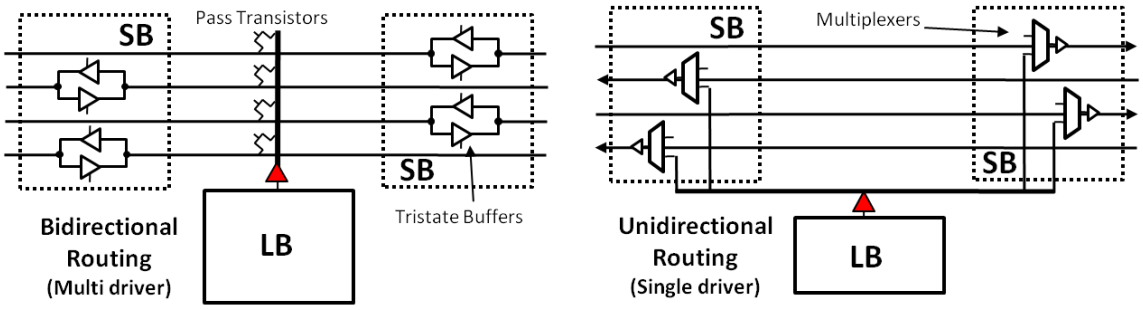


Fig. 2.1b: FPGA architecture fundamentals: Uni/Bi directional routing [Lemieux04, 4.19]

2.1.2 FPGA vendors

In this section the advancements of logic block, the routing architecture and the post 90nm challenges of power are discussed. As it is impossible to discuss effectively the state of the art in a limited scope of a chapter, only very brief global points are discussed that are used as inspiration in the thesis work explained in later chapters. The examples taken are from the latest literature of Xilinx and Altera which provide detailed in depth idea of architectural challenges in most modern FPGAs that currently exist. Additional in depth study of FPGA vendor's patents will be discussed in section 2.3.1 (appendix A1), helping understand the complexity of state of the art solutions and challenges that they address.

1- The Logic Block

Figure 2.2 shows the BLE equivalents of modern devices of Xilinx and Altera. It can be seen that the fundamental concept of LUT-FF remains universal, however lot of enhancements have been made to provide more capabilities of doing more complex tasks. One example is carry chain logics, that facilitate creation of adders/subtractors more efficiently compared to implementing them using LUTs. Another critical enhancement is making the LUT other than logic being also capable of a small memory block and shift registers which significantly saves Flip-Flop usage to do the equivalent job. Figure 2.3 shows a detailed diagram of SLICEM (equivalent of logic block) of 40nm Virtex6 [Xilinx09, 1.3]. It is a cluster of four 6 input LUTs. The significant new changes in this architecture compared to several previous generations is doubling the number of FF to LUT ratio for more effectively implementing heavily pipelined functions and support for clock gating for addressing dynamic power reduction.

LUT Size

The LUT size is one of the most important architecture parameter in FPGAs, as shown in figure 2.4 [Altera06, 1.20] the LUT is typically composed of configuration memory (mostly SRAM) LUT-mask and a set of multiplexers that select the configuration bit to drive output. The figure shows a LUT4 and also shows how LUT4 can be made by using two LUT3 and 2:1 multiplexer. It can be easily observed that silicon foot-print of LUT exponentially increases by increase of its size, for instance LUT6 will occupy more than four times the area of LUT4. LUT4 has remained a dominant choice among FPGA vendors for a long time and still now most FPGAs are LUT4 based. However a bigger LUT size has benefits in terms of speed as more logic is packed in a single LUT hence less global routing will be used when applications are mapped. Intuitively that also provides benefits in terms of power consumption, and finally the area of FPGA is highly dominated by interconnect compared to LUT size so changing the LUT size does not have a huge change in total area (chapter 4 will address these issues in more detail). For such reasons it is observed that both Xilinx and Altera have preferred the move to larger LUT sizes in their newer devices in different ways which is described below along with mapping efficiency challenges of bigger LUT sizes.

Figure 2.5 shows the tradeoff of speed and cost for different LUT sizes. From logic mapping point of view, it can be observed that LUT6 does not have a high mapping efficiency as illustrated in figure 2.6. It can be seen that when targeting a LUT6-only architecture, more than half of the LUT6 are never fully utilized. Starting from Virtex 5 Xilinx moved to full LUT6 from LUT4 with some flexibility of using the LUT6 for implementing two logic functions with some shared inputs (in fig. 2.3 it can be seen that the LUT6 has two outputs). Altera has adopted a more adaptable approach in its Stratix series architecture to obtain a better tradeoff, they call it the Adaptive Logic Module (ALM) [Altera10, 1.14][Altera07, 1.18]. Instead of being a full LUT6 it is composed of smaller heterogeneous LUT sizes as illustrated in figure 2.7. The architecture can adapt to multiple configurations as shown in figure 2.8. This adaptability provides some superior benefits compared to a fix sized LUT6 of Xilinx as demonstrated in figure 2.8.

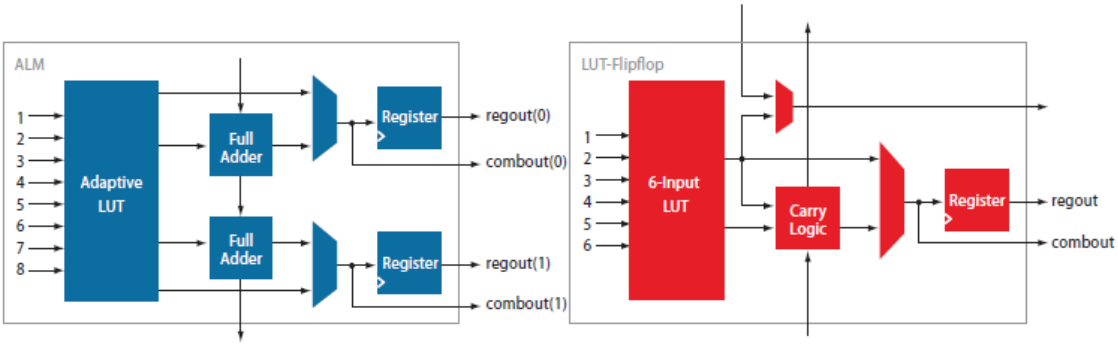


Fig. 2.2: The logic blocks (equivalent) of Stratix III and Virtex 5 [Altera, 1.15]

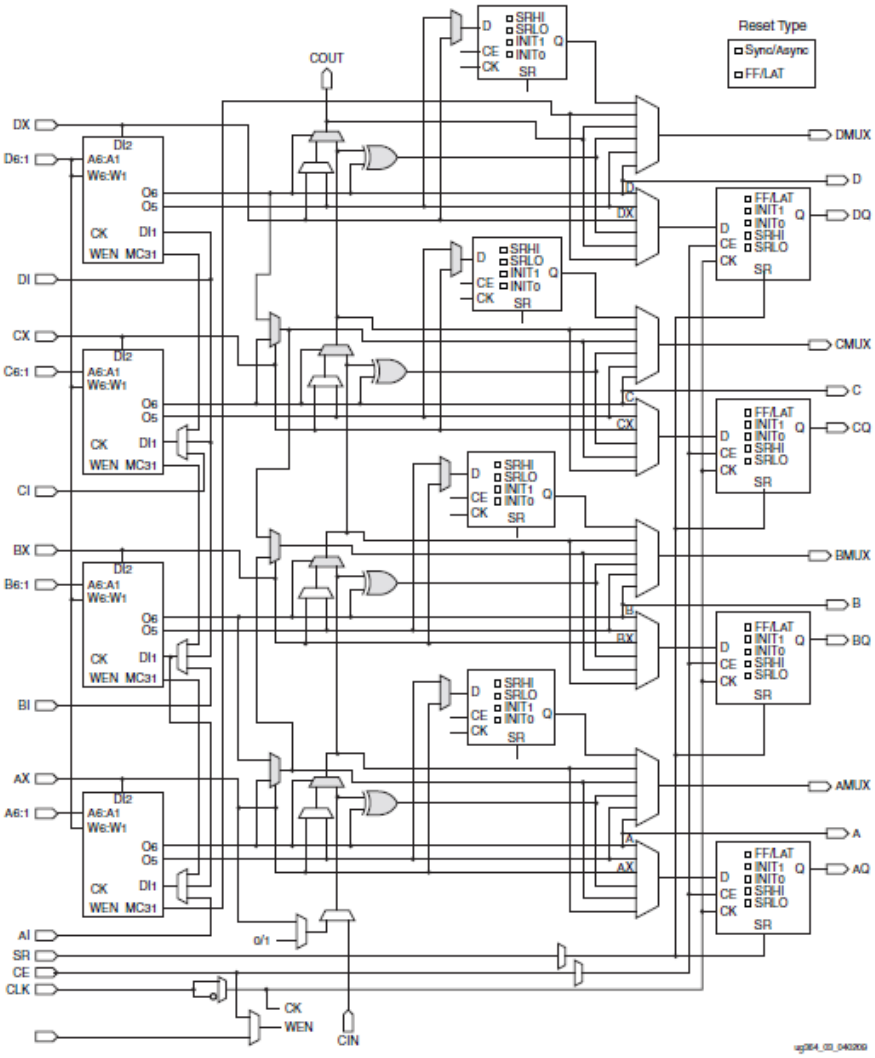
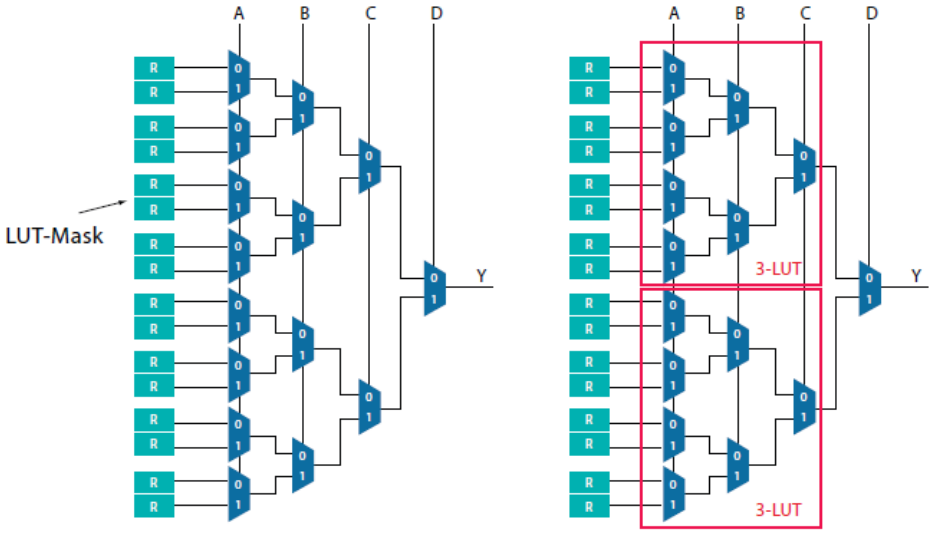


Fig. 2.3: The SLICEM of Xilinx Virtex6 [Xilinx, 1.3]



a'b'c'd' + abcd + abc'd' = 1000 0000 0000 1001 = 0x8009

Fig. 2.4: Building a LUT [Altera, 1.20]

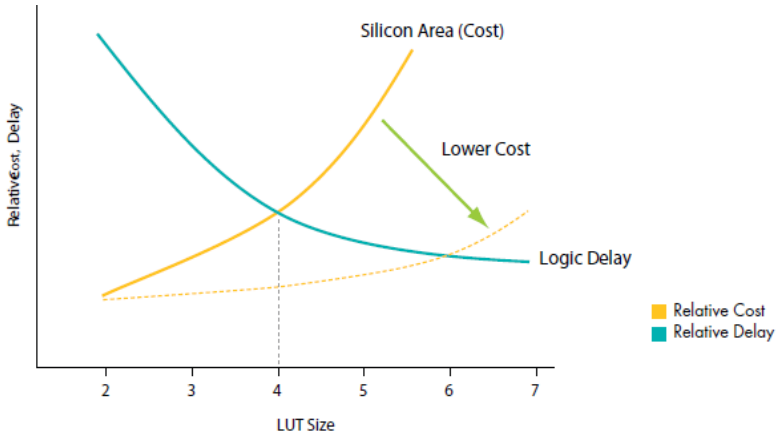


Fig. 2.5: Delay-Cost Tradeoff with LUT Size [Altera, 1.20]

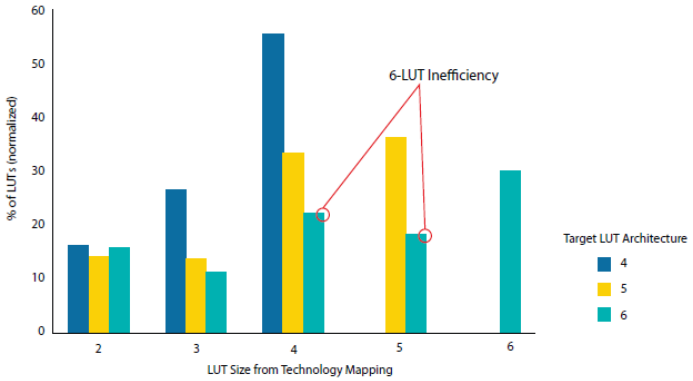


Fig. 2.6: The mapping efficiency of LUT6 [Altera, 1.20]

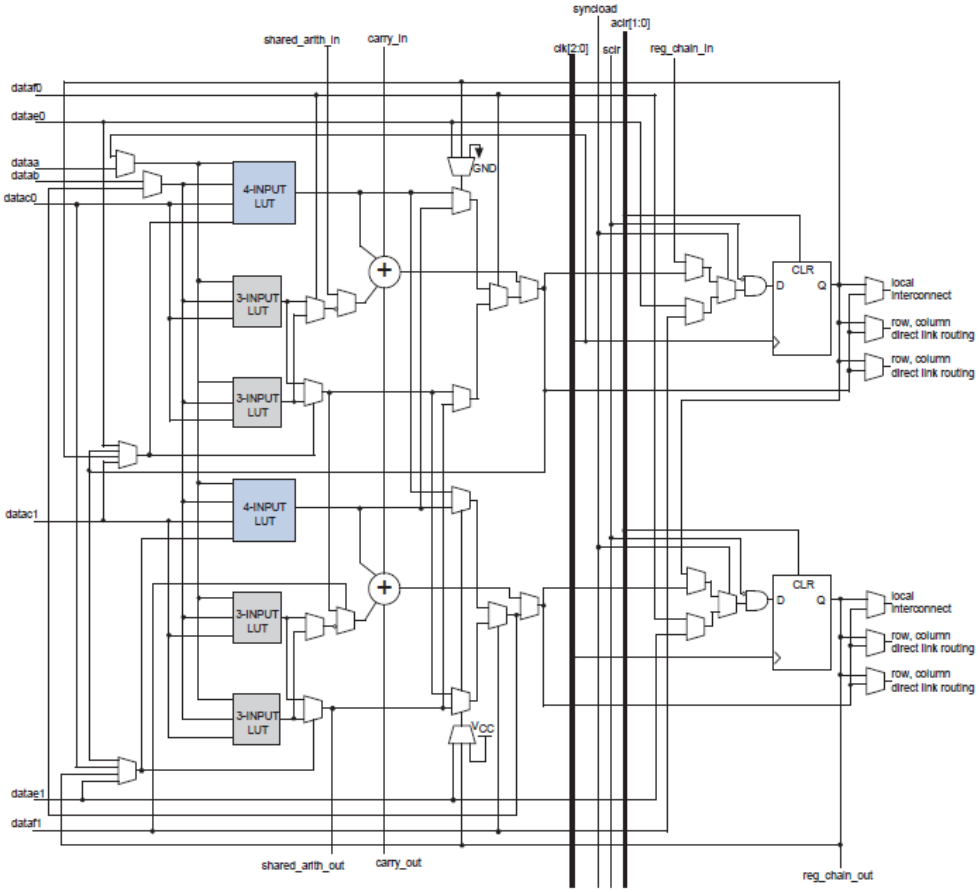


Fig. 2.7: The Detail of Altera StratixIV ALM [Altera, 1.14]

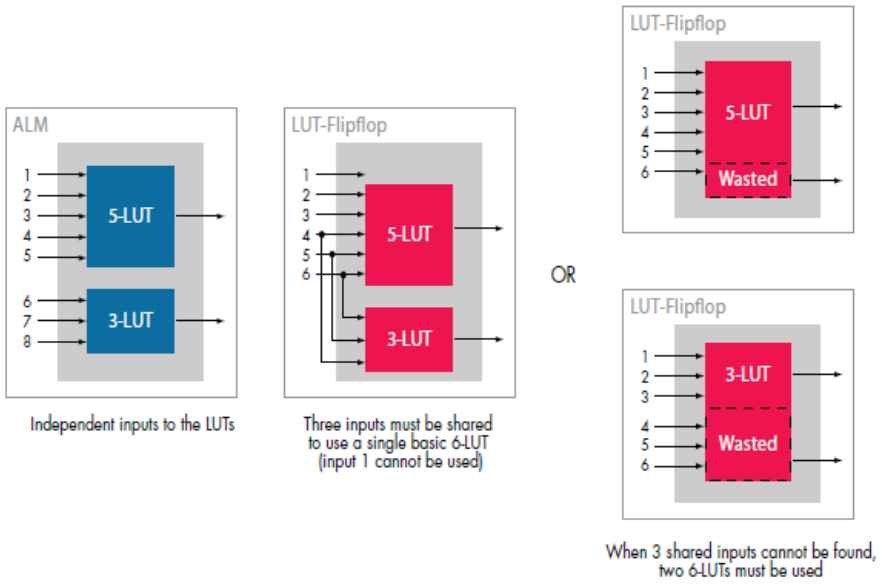


Fig. 2.8: Implementing 5 and 3 input functions on Altera ALM and Xilinx LUT6 [Altera, 1.20]

2- Routing Architecture

The most challenging and expensive part of an FPGA is the routing architecture that connects all the elements of FPGA. All FPGA vendors have architectural differentiations that distinguishes their product from competitor in some or special aspects. Figure 2.9 shows the abstract block diagram of Xilinx VirtexII devices [Xilinx07, 1.8]. It can be seen that all the fundamental logic (CLB) elements and specialized blocks like memory, multiplier, I/Os etc. are arranged in mesh style architecture/Island-Style as was discussed in introduction. Figure 2.10 illustrates the logic block surrounded by sea of routing resources [Altera07, 1.15]. This topology is common among majority of FPGA vendors; it is prominent and particularly favored due to its layout friendliness and scalability. The routing architecture in most cases can be divided into three fundamental parts (SB, CB and routing channels) as was discussed above. They are briefly revisited below for state of the art architecture comparison discussions.

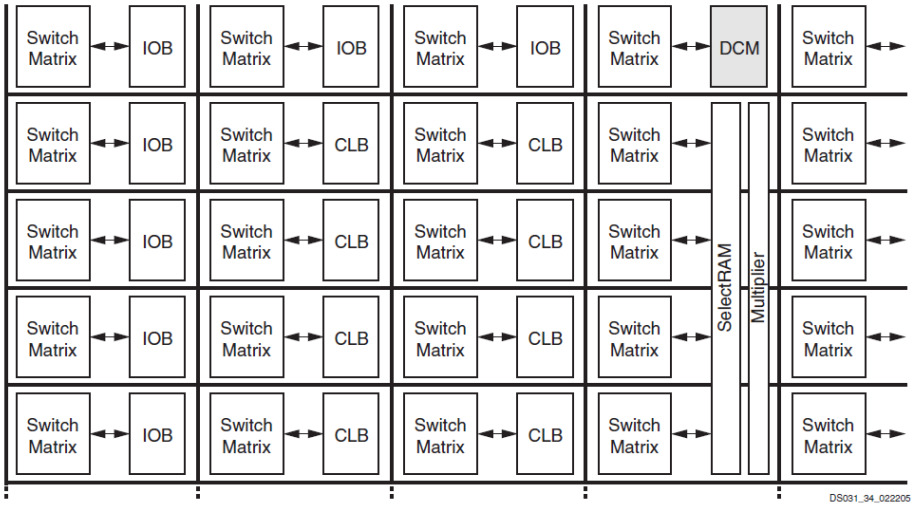


Fig 2.9: Global Architecture of Xilinx Virtex-II [Xilinx, 1.8]

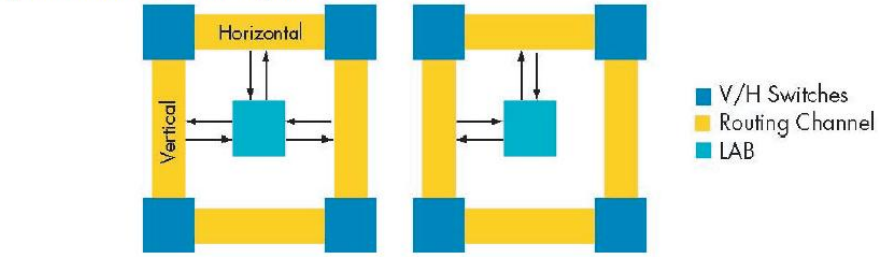


Fig. 2.10: Routing Architecture sides Altera (left), Xilinx (right) [Altera, 1.18]

Switch Block & Connection Block

The switch block performs the function of main switching hub of the routing channels. Its immense amount of routing switches/muxes provides the programmable interconnection of the routing tracks for the routing of the target application. In figure 2.10 (V/H switches) the basic principle is illustrated, showing the switch block serving as an interchange of traffic on horizontal and vertical channels.

Connection block is closely linked with switch block and the logic block. From a technical stand point connection block is the switching component that connects the logic block to the routing tracks/channels. Figure 2.10 illustrates that, where the concept is shown for Altera Stratix series (left) and Xilinx Virtex 5 devices (right) [Altera07, 1.18]. Altera connects (connection blocks) logic block to three sides of routing channels compared to two sides by Xilinx. Depending on the style of implementation of FPGA architecture, connection block can have different forms; it is possible to merge it with switch block (concept remains same) as seen in figure 2.9 of Xilinx style architecture where switch block performs all the switching activity.

Routing Channels & HOP

The third fundamental element of routing architecture is the routing channels. These are physical metal lines (distributed on several metal levels), providing the data highway of the architecture. Their size/width (number of tracks) varies from application to application and in general set to the value high enough to allow FPGAs able to route industrial benchmarks [Altera05, 1.22].

The routing channels have significant influence on the performance of the mapped application. The challenges are two folded. First the routing channels are very expensive on silicon and their size (switch block and connection block size is also linked with them) greatly changes the silicon foot-print of the device which also indirectly influences on power consumption and speed. So the switch block and connection block architecture is big research challenge to efficiently use the channels. Secondly like general problem with automotive in-city, inter-city traffic etc., heterogeneous length routing channels are common in modern FPGAs (long wires) which provide different kind of highways for data traffic in FPGAs to have better speed and less congestion. Figure 2.11 shows the hierarchy of routing resources of Xilinx Virtex-II FPGA. Interesting point to note is that there are more long wires in the architecture (Long, Hex, Double) compared to direct wires to neighbors. The Altera Stratix series architecture is also composed of different length routing tracks including 4, 8, 16 and 24 [Altera06, 1.20]. Figure 2.12 shows how the logic block of Stratix IV (LAB) are connected to the multi length routing tracks in three directions as was shown in figure 2.10. The crossing of the routing tracks with multiple crosses in top middle of figure 2.12 represents the switch block that allows switching between different tracks.

The pattern of the long wires in the global architecture is generally referred as a HOP (jump, long jump) in literature which explains how far a signal go from one logic block in a single HOP (jump). Figure 2.13 shows the hop patterns of some latest devices of Xilinx and Altera. Starting from Virtex-5 Xilinx added several diagonal long tracks compared to Virtex-4 allowing a much wider access to the logic blocks (CLBs) in 2 HOPs. It can also be observed that for a single HOP Altera (right) has more logic blocks reachable compared to Xilinx and its HOP pattern has a more custom shape compared to regular and more expensive Virtex-5 pattern [Altera07, 1.23][Altera06, 1.20]. Chapter 4 will further address HOP in architectural explorations experiments.

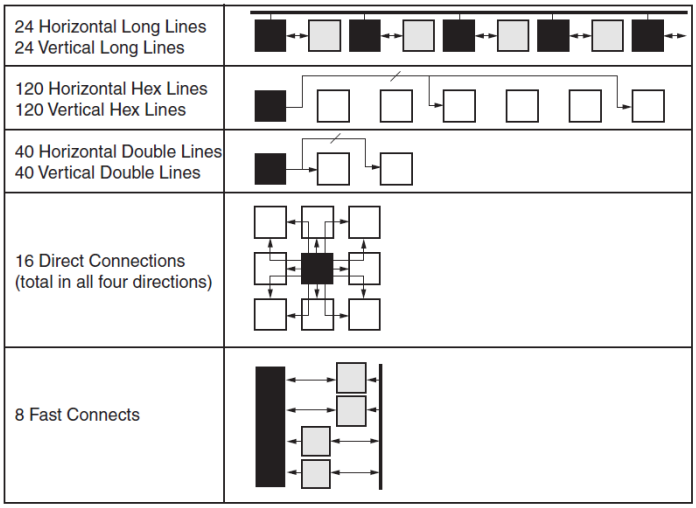


Figure 49: **Hierarchical Routing Resources**

Fig. 2.11: Hierarchical Routing Resources of Virtex-II [Xilinx, 1.8]

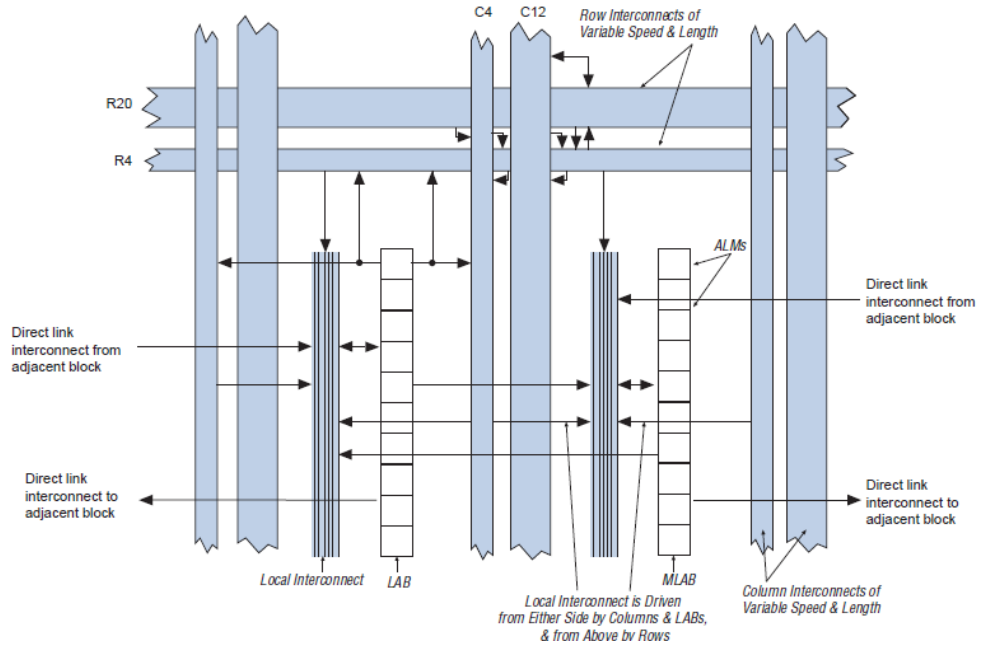


Fig. 2.12: Logic Array Block (LAB) structure of Altera Stratix IV [Altera, 1.14]

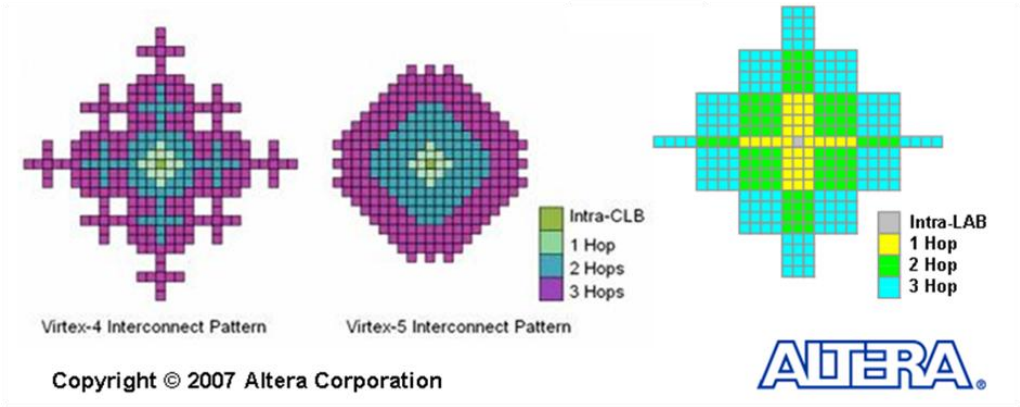


Fig. 2.13: The routing HOP [Altera, 1.23]; {Note: source image is blurry!}

3- Architectural Heterogeneity

The huge power of FPGAs of prototyping virtually any digital circuit due to their Fine-Grain flexibility of LUTs comes at a heavy silicon price. To address this issue, over time FPGAs got more and more heterogeneous by adding Hard Blocks of logic/memory [Altera_Betz10, 2.4] allowing bigger circuits to be implemented on them. The first elements came were memories in around mid 90s followed by multipliers and the trend went on and modern FPGA now contain multitude of heterogeneous blocks like DSPs, PLLs, memory controllers, high speed serial IOs and even Microprocessor cores. Figure 2.14 shows the abstract block diagram of 40nm Stratix IV device, similar properties are found in equivalent Xilinx devices. The feeling of high level of heterogeneity of modern FPGAs can be seen in it.

Column Based Architectures

Figure 2.15 shows the more elaborated view of this (figure 2.14) device showing how the fine grain logic (LEs) is mixed with general purpose coarse grain hard blocks (memory, multipliers, etc) in a column fashion. This is important point to note, *like Island style architecture is common in most FPGAs, putting the hard blocks in columns is also very common* among FPGA vendors as it provides better layout efficiency and ease of connection to route architecture compared to randomly placing them.

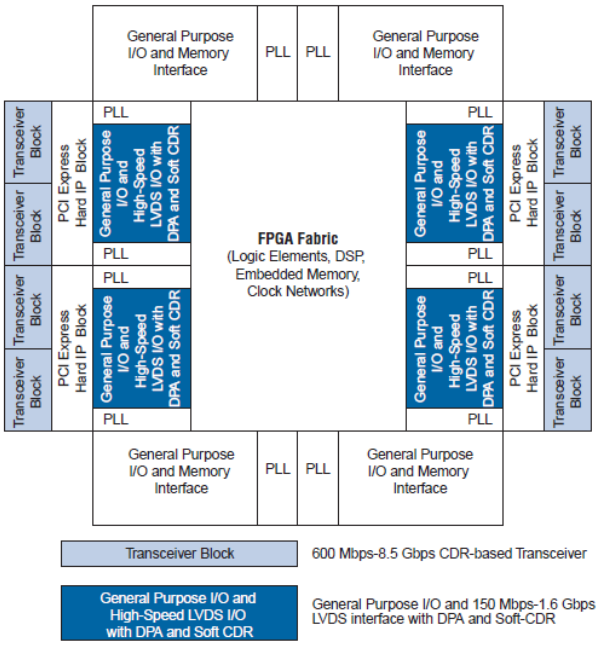


Fig. 2.14: Altera StratixIV block diagram [Altera, 1.14]

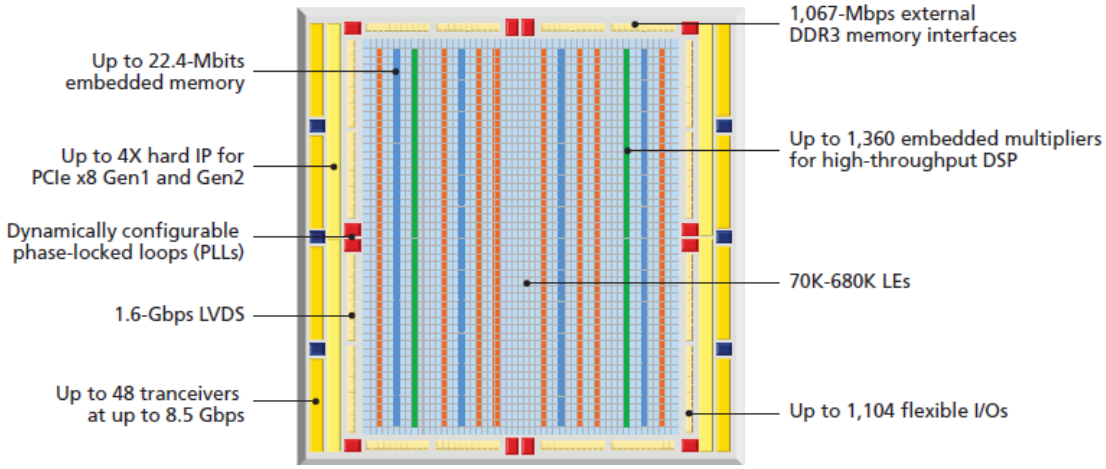


Fig. 2.15: Stratix IV architectural elements with columns of hard blocks [Altera, 1.14]

4- Post 90nm Challenges of Power vs. Performance

Starting from 90nm the whole industry was struck with the issues of power consumption (static in particular) which has now become the paramount challenge for all semiconductor companies. For FPGAs it is further crucial as they already have a hard challenge of power compared to ASICs/ASSPs and as the value proposition of FPGAs is highly tied to Moore's law for more logic density, they have to keep pace with the latest node. The issue of power has added an additional dimension in the challenges of FPGA research other than classical well-known challenges described above. Understanding their fundamentals is essential for finding research motivations.

Both Xilinx [1.4][1.5][1.6][1.7] and Altera [1.16][1.19][1.23] have done numerous architectural and silicon innovations specifically to address power. A brief overview of key issues and how they have innovated to deal with these issues is presented below.

Static Power issues

Figure 2.16 shows the exponential growth of static power over past few generations from being almost negligible to a significant value. 90nm is widely considered as an inflection point when industry observed the dominant issue of leakage power approaching dynamic power and has become a major design challenge for chip design in beyond 90nm nodes. Figure 2.17 shows the main components of leakage power and figure 2.18 illustrates them and design techniques that address them. The two key components of leakage are the gate leakage and source-drain leakage. As technology shrinks to new node for higher density (usually 2x of Moore's law) represented by the mathematical formula shown in figure 2.19 [Xilinx09, 1.4], the channel length and gate oxide thickness decreases hence increasing the leakage. Furthermore source to drain leakage exponentially grows with temperature compared to gate leakage as illustrated in figure 2.20 [Xilinx05, 1.7].

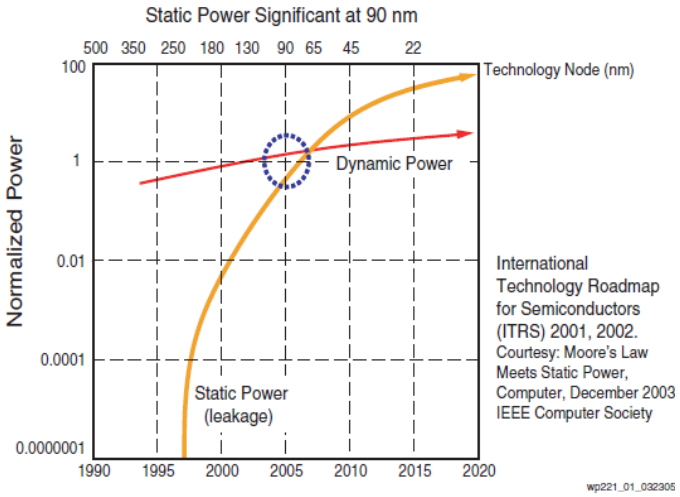


Fig. 2.16: Static & Dynamic Power vs Tech. Node [Xilinx, 1.7]

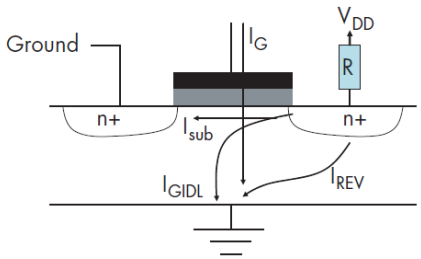


Fig. 2.17: The Components of leakage current [Altera, 1.16]

Source	Impact	Sensitivity	Design Techniques
Subthreshold (weak inversion) leakage (ISUB)	Dominant	Supply voltage Gate threshold voltage Temperature Channel length	Reduced core voltage Increased voltage threshold Increased gate lengths
Gate-induced drain leakage (IGIDL)	Small	Gate oxide thickness Supply voltage	Dual gate oxide
Gate direct-tunneling leakage (IG)	Small	Gate oxide thickness Supply voltage	Dual gate oxide
Reverse-biased junction leakage current (IREV)	Negligible	N/A to low voltage CMOS	None required

Fig. 2.18: The Leakage current Impact, sensitivity and Design techniques [Altera, 1.16]

$$\textbf{\textit{Transistor density Scale factor}} = \left(\frac{\textbf{\textit{Previous Node Size}}}{\textbf{\textit{New Node Size}}}\right)^2$$

Fig. 2.19: The transistor density scale factor [Xilinx, 1.4]

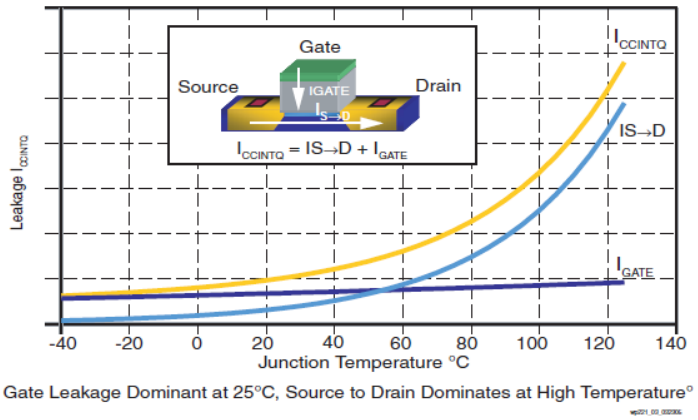


Fig. 2.20: Leakage Power vs Temperature in 90nm Virtex-4 [Xilinx, 1.7]

Innovations by Xilinx & Altera

To address issues of static power both Xilinx and Altera have taken some similar and some differentiating approaches. The gate leakage is mostly addressed by using multiple gate oxide thickness for different transistors depending on their criticality for timing. Xilinx used triple gate oxide [1.5][1.7] to address the issue starting from Virtex4. On 28nm node both Xilinx and Altera are switching to HighK Metal Gate (HKMG, that was first introduced and pioneered by Intel for its 45nm Hafnium innovation) technology of TSMC [Altera_Betz10, 2.4][Xilinx_Lysaght10, 2.5].

The source to drain leakage is related to threshold voltage and channel length (figure 2.18). In this regard both Xilinx and Altera have adopted different styles. Figure 2.21 shows the Programmable Power concept of Altera [1.16][1.19]. The place and route (PAR) tool adaptively assigns different threshold voltage to the logic blocks based on their timing criticality. The logic on critical path is set with low threshold voltage making the transistors to have high speed at expense of higher leakage (power vs performance) and non critical blocks are put to low leakage and low speed hence significantly saving static power. Compared to dynamically changing threshold voltage approach of Altera, Xilinx adopted to statically assign different threshold voltages to different transistors and in 40nm Virtex6 has exploited channel width also to address the issue. Figure 2.22 shows four primary types of transistors in Virtex 6 device having very few high leakage high speed 40nm transistors at the expense of area (power vs area) by using majority of transistors at 44nm [Xilinx09, 1.4].

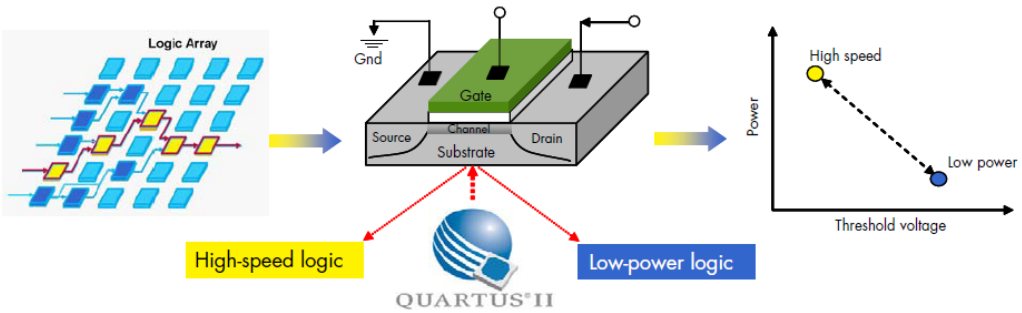


Fig. 2.21: Programmable Power Technology of Altera through Quartus II [Altera, 1.16]

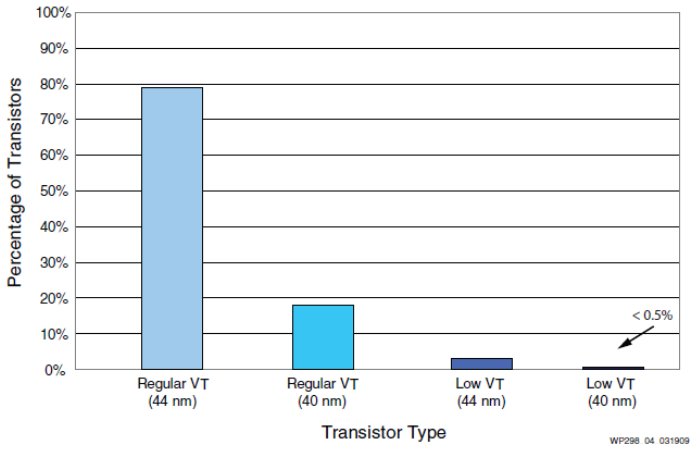


Fig. 2.22: Transistor type distribution in Virtex-6 FPGAs for combating power [Xilinx, 1.4]

Power & Speed scaling vs Moore's law

From figure 2.23 and 2.24 it can be observed how challenging it is becoming to obtain double performance and power gain like in area when migrating to next nodes starting from 90nm. It can be seen that numerous architectural innovations have led to decrease the power in beyond 90nm Xilinx devices however the sharp classical benefits of going to next node have greatly decreased. For instance moving from 130nm to 90nm yielded in around 40% reduction in static and dynamic power. Moving further down it took almost two generations to get around 40% dynamic power decrease and static power is further lagging it but is decreasing due to new innovations. Comparing from figure 2.17 it shows how important the static power innovations are to move ahead. Similar issues regarding speed can be observed, high gain in speed from 130nm to 90nm Altera devices decreases in comparison going to 65nm. Such challenges are making harder to get full benefit of increased logic density of Moore's law and power is emerging as the primary concern compared to area and speed in the past (trade power vs area).

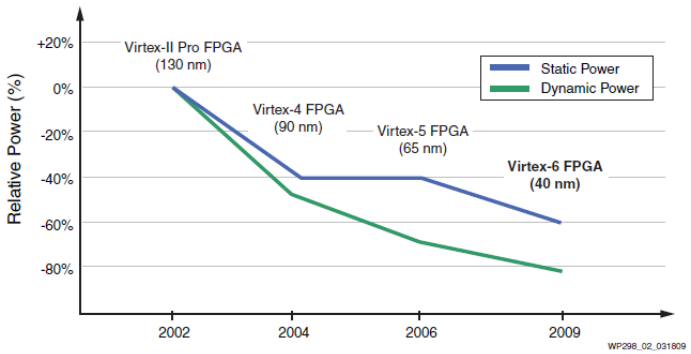


Fig. 2.23: The relative power consumption of Virtex FPGAs on different nodes [Xilinx, 1.4]

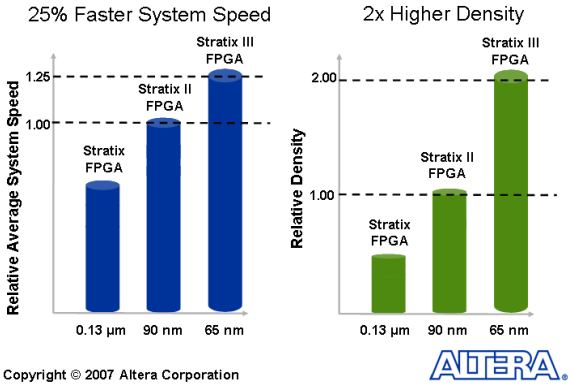


Fig. 2.24: Performance and Density Improvement in Stratix FPGAs on different nodes [Altera, 1.23]

2.1.3 Academics

The research of FPGA architecture is closely coupled with the FPGA CAD. Not only the CAD tools are needed to perform mapping of applications on an FPGA but also the exploration of new architectures and concepts is highly dependent on the quality and level of exploration freedom the tools provide to FPGA architect and researchers. Section 2.1.2 provided a comprehensive overview of advancement of FPGA architectures in leading FPGA vendors. Unfortunately it is difficult to analyze/get research benefit in a similar way from the CAD tools of state of art as they are proprietary and highly focused on the products of the respective companies.

The FPGA CAD research is not the main contributing part of this thesis work (it was done by Menta's CAD team in close link with this thesis), however understanding the fundamentals is mandatory for any FPGA research as knowing basic fundamentals of hardware is essential for FPGA CAD researchers. In this regard a good help was obtained by the most recognized and highest cited academic research in FPGA CAD and architecture from University of Toronto's work of VPR [Betz&Rose97, 4.12][Betz&Rose99, 4.1] which has a de-facto status in FPGA research in academics. This not only gave insights of FPGA architecture and CAD challenges but also helped to understand the research work of many other scientists that is based or partially based on VPR.

In this section first a brief overview of the FPGA CAD is presented, identifying the major steps for application mapping on FPGAs. Second a comprehensive overview of well recognized academic research addressing several key aspects of FPGA architecture during last ten years is discussed to get research motivations and finding room for new contributions.

1- FPGA CAD

Figure 2.25 shows a generic flow of FPGA CAD for mapping an application described in high level HDL (e.g. VHDL, Verilog) down to configuration streams that program the target FPGA. It is interesting to note that the flow is same/almost same for all the FPGAs. All the steps of flow are generally independent (no cross iterations) or are independent for most of the part (the works discuss here and in the thesis work). However the CAD tools of state of art can have more complex interlinks among different phases with several interactions and iterations among different phases for reaching a more optimal solution (e.g. physical synthesis, combined packing + placement + routing for timing closure etc.) [Cong06, 4.41].

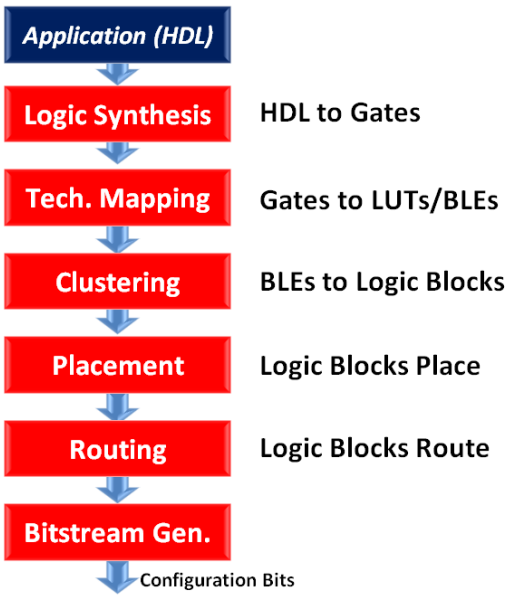


Fig. 2.25: Generic FPGA CAD flow [Betz&Rose99, 4.1]

The phases/steps of the flow are briefly described below. Routing is discussed in a bit more detail as understanding the concept of Routing Resource Graph (RRG) is essential for the exploration works and tools designed in this thesis. The deeper details are beyond the scope of this thesis. A more detailed overview of them can be found in [Betz&Rose99, 4.1][Cong06, 4.41].

Logic Synthesis

The synthesis step transforms the circuit description mostly written in HDLs (e.g. VHDL, Verilog or mixed) into a network of Boolean logic consisting of gates and flip-flops/latches (and datapath operations like multipliers, adders, memories etc. for heterogeneous architectures). This step is usually technology independent (not always in state of art) with no FPGA specific optimizations done to the logic. The synthesizer optimizes the Boolean logic to have an optimal netlist of combinational and sequential elements (gates).

Technology Mapping

The technology mapping phase maps the output of synthesis to dedicated circuit structures of target device like on-chip multipliers, adders with dedicated carry-chains and embedded memories for optimized datapaths. The optimized control logic is mapped to logic blocks (BLEs as was discussed in the introduction). The datapath operations can be mapped to BLEs as well (universal prototype power of BLEs) if the dedicated circuit structures of Hard Blocks are not available or not convenient to use [Cong06, 4.41]. The tools like FlowMap [Cong94, 4.46], SIS & ABC (Berkeley University work of Brayton & Mishchenko) from academics are prominent examples of technology mapping (on BLE only mostly) which are widely used in academic research on FPGAs. The output of mapping is a netlist of LUTs and Flip-Flops/Latches.

Clustering/Packing

In more advanced tools this step is often combined with placement [Cong06, 4.41]. The clustering phase has/can have two main steps. First it packs the LUTs and FFs in the technology mapped netlist into BLEs. Depending on the sequential or combinational output of LUT in the netlist the 2x1 output multiplexer selects the required outputs. In state of the art, the BLE is more complex allowing simultaneous use of combinational and sequential output of LUTs and more complex FF structures as was discussed in state of art FPGAs overview.

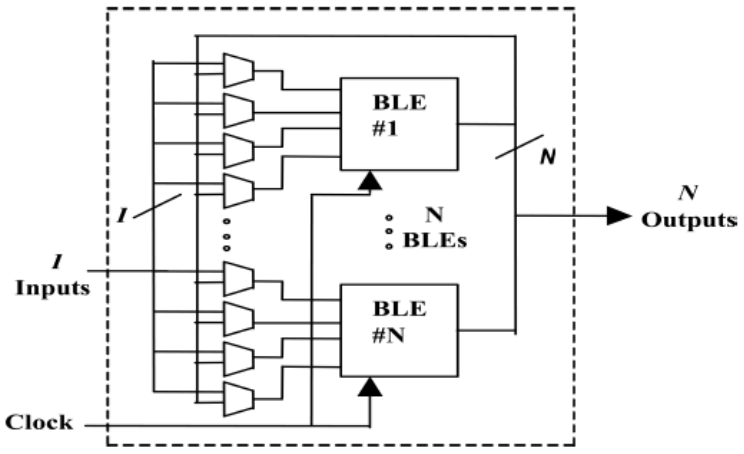


Fig. 2.26: Fully Connected Clustered Logic Block [Betz&Rose99, 4.1]

The second step of clustering groups BLEs into clusters/logic blocks as shown in figure 2.26 depending on number of BLEs (cluster size, N) in the logic block of target FPGA. The cluster block shown in figure

2.26 is a fully connected cluster [Betz&Rose99, 4.1] that makes all the inputs (I) and outputs (N) of the block logically equivalent providing the router added flexibility for routing. One of major optimization goal of clustering is to cluster BLEs in such a way that the inter-communication between the clusters is minimized which allows lesser use of global routing resources and hence added speed of implemented circuit. The output after clustering (for homogeneous FPGAs) is a netlist of logic blocks that are placed and routed by the final steps. The VPack and T-VPack [Betz&Rose99, 4.1] are widely used academic tools in research for clustering the technology mapped netlists.

Placement

In the placement phase the algorithms determine which physical logic block within an FPGA should implement the corresponding logic blocks required by the circuit. The optimization goals are to place the connected logic blocks close together to minimize the required wiring (wire length-driven placement), sometimes place the blocks to balance the wiring density across the FPGA (routability-driven placement) or to maximize the circuit speed (timing-driven placement). Placement has a significant impact on the performance and routability of the mapped application. Among different types of placements are partitioning based, analytic, embedding-based and simulated annealing-based [Betz&Rose99, 4.1][Cong06, 4.41].

Routing

Routing is one of the most basic, tedious, yet important step in FPGA design flow. It is the last step in the design flow prior to generating the bit-stream to program the FPGA. FPGA routing is similar to the general ASIC problem in terms of the objectives (need to successfully connect all signal nets subject to timing constraints). However, FPGA routing is more restricted in the sense that it can use only the prefabricated routing resources available on the target device; therefore to achieve 100% routability is more challenging in FPGAs [Cong06, 4.41].

To model all the available routing resources in an FPGA, a routing resource graph is created as an abstract data representation to be used by the routers. Given an FPGA architecture, the vertices in the routing-resource graph represent the input and output pins of the logic blocks as well as the wire segments in the routing channels. The edges represent the programmable switches that connect the two vertices. A unidirectional switch, such as a buffer, is represented by a directed edge, while a bi-directional switch, such as a pass transistor, is represented by a pair of directed edges. To model the equivalent pins, a source vertex connects to all the logically equivalent output pins of a logic block, and a sink vertex connects from all the logically equivalent input pins of a logic block. Figure 2.27 shows an example of a routing-resource graph for a portion of an FPGA whose logic block contains a single two-input, one output LUT. In general, a node may have a capacity that indicates the maximum number of nets that can use this vertex in a legal routing. In shown example, the source vertex has capacity one, while the sink node has capacity two. Modern FPGAs have tens of thousands of logic blocks; the routing resource graph can be very large. Its generation is typically done automatically by a software program, which models the given FPGA, builds the routing-resource graph for a basic tile of the architecture, and then replicates the graph many times and stitches them all together to form the routing resource graph for the entire FPGA [Cong06, 4.41].

In many cases, one needs to build the placement and routing tools for an FPGA under development in order to provide quantitative evaluation of the choice of various architecture parameters before finalizing the FPGA architecture. In this case, one needs to generate a routing-resource graph from a set of architecture parameters, as the real FPGA model is not yet available. The typical set of parameters needed for routing include [Betz&Rose99, 4.1]:

a) Number of logic block's input and output pins.

b) Side(s) of the logic block from which each input and output is accessible.

c) Logic equivalence between the various input and output pins.

d) Number of I/O pads that fit into one row or column of the FPGA.

e) Relative widths of the horizontal and vertical channels.

f) Relative widths of the channels in different regions of the FPGA.

g) Switch block topology used to connect the routing tracks.

h) Fc values for logic block inputs and outputs, as well as I/O; pads (Fc represents the number of routing tracks in the channel that each input or output pin connects. The Fc value may vary for an input pin, an output pin, or an I/O pad).

i) Wire segment types and distributions: for each segment type, one needs to specify segment length, fraction of tracks in the channel with such type, type of switches, and population of the switches on the segment, etc.

Parameters (a) to (f) are needed for global routing, and additional parameters (g) to (i) are needed for detailed routing. A good routing resource generation tool should be able to:

- Detect any inconsistency in architecture parameter specification
- Provide reasonably good assumptions of the missing parameter in case of partial architecture specification (which is quite common in the early stage of architecture exploration).

One important contribution of the VPR placement and routing tool is that it provides a simple language for the user to specify a reasonable set of architecture parameters for an FPGA under investigation and generates the corresponding routing-resource graph automatically [Betz&Rose99, 4.1][Betz&Rose00, 4.11].

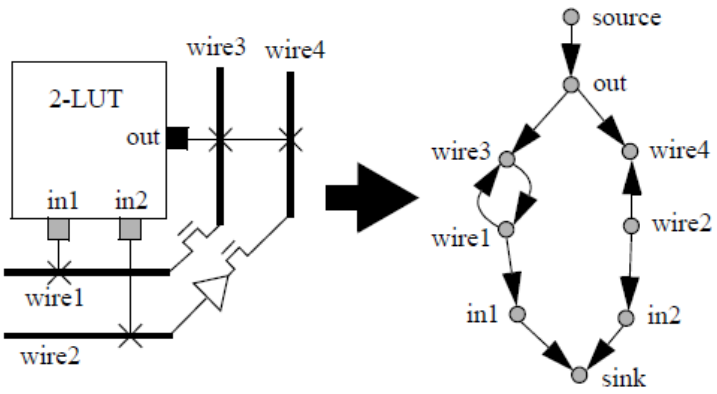


Fig. 2.27: Modeling FPGA routing as a directed graph [4.10]

Bitstream Generation

It is the final step of the flow. It takes as input the mapping, placement and routing information and generates the bitstream to program the configuration cells of LUTs, routing, IOs, Hard Blocks etc. to implement the mapped application on the target FPGA.

2- Architectural Research

This section provides brief overview of prominent contributions from academics for addressing fundamental research challenges of FPGAs. A good global overview on advancements and challenges in FPGA architecture can be found in [Kuon.Tessier.Rose08, 4.2] and of FPGA CAD in [Cong06, 4.41].

LUT Size & Cluster Size

The first fundamental question that comes in every FPGA researcher mind is what the best LUT size is for an FPGA. Is there a magic LUT size (2, 3, 4, 5, 6, 7 etc.) that can be best global choice for FPGAs. The answer is bit complex; one cannot just compare two FPGA architectures based on the LUT size they use. It depends a lot on the architecture style of FPGA, the way LUTs are used in that and the properties/quality of CAD tools that map applications, which leads to final properties/capabilities of that FPGA. It also depends on the target requirements for that FPGA (speed, area or power optimized etc.). However LUT size of 4 since the early 90s and even now is the most widely used LUT size among most of the FPGAs providing a nice tradeoff among major criteria like area, power, and speed efficiency.

LUT size is often investigated along with the cluster size as both have an impact on each other. The effect of LUT and cluster size was addressed in detail for a fully connected cluster in [Betz&Rose99, 4.1] [Ahmed&Rose04, 4.8][Marquardt.Betz.Rose00, 4.9]. They indicated that for best area and delay tradeoff the LUT size from 4-6 and cluster size of 3-10 are the best choice. [Cong06, 4.41] shows work done to address the clusters with heterogeneous LUT size like found in Altera's Stratix series as was observed in state of the art discussions section.

Routing Architecture

Routing is the most challenging and exciting part of FPGA research. It has several areas to address such as architecture topologies, SB patterns, heterogeneous wire lengths, uni/bi-direction routing, and silicon implementation issues etc.

One of most important aspect of VPR tools suite is its ease of creation of routing architecture at a higher abstraction level using some fundamental properties given in architecture description file (as was discussed in last section) [Betz&Rose99, 4.1]. This allows experimenting different architectures with ease of creating them through VPR. Original VPR [Betz&Rose97, 4.12] was focused on bidirectional routing architecture which was common in FPGAs until early 2000s. Modern FPGAs use unidirectional routing (as was discussed in fundamentals section in beginning). The work of [Lemieux04, 4.19] explains the benefits of unidirectional routing over bidirectional. VPR 5 [Luu.Kuon.Rose09, 4.5] has updated the possibility to use unidirectional routing along with better architecture description mechanism using XML and support for basic hard blocks. The architecture of Switch Block (SB) has an impact on the routing efficiency of the routing architecture. [Wilton97, 4.25] proposed a new kind of SB (Wilton) with added routability due to the diversity it offers to route among channels when they cross a SB. [Imran&Wilton99, 4.35] enhanced the Wilton switch for multi-length routing architectures. The work [Lemieux&Lewis02, 4.21][Lemieux&Lewis02, 4.22] addressed the issues of diversity of SB and explores the benefits by experimenting different architectures. Chapter 4 will address these issues and SB in detail.

Routing architecture has another challenge that is hard to address by classical hardware innovations; applications mapped on the FPGA have different requirements of routing. Some require normal some low and some very high (chapter 4 will address them in more detail), however once the device is fabricated one cannot change the routing architecture. [Tom.Leong.Lemieux06, 4.18] addresses the issue by software at the expense of using more logic blocks than needed (often possible in real cases) to route the un-routable designs by depopulating the logic blocks in congested areas.

The physical silicon implementation of the architecture also has a significant effect on the characteristics of the architecture. [Betz&Rose99, 4.1][Kuon&Rose09, 5.2][Kuon&Rose08, 4.6][Kuon&Rose08, 4.7][Lemieux&Lewis02, 4.23] [Lee.Lemieux.Mirabbasi08, 4.16][Lemieux04, 4.19] have addressed several issues of transistor sizing, buffers, multiplexer designs etc.

Power Consumption

Power consumption is the newest and hottest challenge at present in industry and academics. Compared to research on classical fundamentals of FPGA that spans around two decades more focused on area and speed, less work exists in this new area, as it has become a hot challenge around mid 2000s. However it will definitely rapidly grow in coming years, as power has become main center of focus among three fundamental pillars of Area, Power, and Speed. Some interesting work can be found in academic research addressing some of the power challenges that were observed in state of the art discussions section 2.1.2. [Anderson05, 4.85][Cong04, 4.42] discusses the challenges of power and addresses the CAD and architectural innovations to reduce both static and dynamic power. [Lee.Chen.Cong05, 4.48][Lee.Chen.Cong03, 4.49] address architecture evaluation based on power by using their framework fpgaEVA-LP and explores the LUT size and cluster size in light of power. [Lee&Cong04, 4.50][Chen&Cong04, 4.51] explores architectures and CAD for the use of multiple supply and threshold voltages to reduce the power. [Hsieh&Cong08, 4.52] proposes use of flip-flops in the FPGA (along with CAD infrastructure to avoid data hazards due to additional pipelining due to FF insertions) to act like a glitch-filtering firewall for decreasing the dynamic power in the routing architecture. [Lamoureux.Lemieux.Wilton08, 4.15] also addresses glitch-filtering issue using hardware innovations by making the data at inputs of LUT arrive at same time by using delay elements with small area overhead. [Chow.Luk.Wilton05, 4.37] discusses the use of dynamic voltage scaling in FPGAs by using logic delay measurement circuit. [Wilton.Ang.Luk04, 4.38] addresses the gain in energy reduction by writing applications on FPGA with more pipelining (guide for FPGA programmer).

FPGAs vs ASICs Gap

A question often come-up in everyone's mind is, how much is the difference between FPGAs and ASICs or in simple words the FPGAs vs ASICs silicon gap. FPGAs have great benefit in terms of programming flexibility compared to ASICs however it is widely known that huge interconnects and fine grain nature of computing model of FPGAs leads to high silicon cost (area), slower speed and higher power consumption. But how much is that difference?, negligible, large or very large remained a curiosity among researchers and industry.

To answer this question in general is really complex; one has to make a borderline or start point to answer this question. To illustrate it a bit further for a given source application there will be multiple ways to implement it on ASICs and similar goes for an FPGA. FPGA is not a universal entity; each device from same or different vendors will/can have different results. Finally the one who has the highest capability and insight to answer this question is the FPGA vendors, but it is straightforward to understand that it is not an easy task for them from commercial point of view to answer these questions openly and fairly!.

[Kuon&Rose07, 4.3] from academics has addressed the FPGA vs ASIC Gap issue and is one of the most inspiring source of knowledge among academics and industry. It has compared the gap on the 90nm node for benchmark applications on standard cells and 90nm Stratix II with co-operation of Altera to give inside to the area figures of their device (such things are mostly closed for public). It is found in their work that:

- Area Gap: around 35X for pure LUT based FPGA (homogeneous FPGA)
- Speed Gap: around 3X-5X
- Power Gap: around 14X , for dynamic Power

These gaps can be reduced by use of hard blocks. This work serves as a motivation for FPGA researchers and vendors about importance of these gaps and there is still so much more to be done to make FPGAs a real competitor to ASICs/ASSPs. This provides a motivation for researchers to investigate heterogeneous FPGAs. Work on this direction is still greatly lacking and at a very basic level in academic research compared to other areas of FPGA that are widely addressed. VPR 5 [Luu.Kuon.Rose09, 4.5] is a step towards heterogeneity that has upgraded original VPR for allowing basic hard blocks experimentation. [Parvez10, 4.53][Parvez.Marrakchi.Mehrez10, 4.54] has also addressed this direction. It has also explored the potentials of Application Specific Inflexible FPGAs (ASIF) for reducing the ASIC vs FPGAs gap by proposing a new kind of middle device that is targeted and optimized for few known target applications.

Different Arch. Styles (other than Island Style)

Majority of the FPGAs are island-style in terms of architectural topology as was discussed before. It still remains the most dominant choice among FPGA vendors due to its layout friendliness. Research is also conducted for some other topologies like Hierarchical/Tree based styles. [Kuon.Tessier.Rose08, 4.2] presents an overview and references to work done in this direction and what are the physical implementation and application mapping issues of hierarchical architecture which makes island-style preferred choice among FPGA vendors. [Marrakchi08, 4.58][Mrabet09, 4.57][Marrakchi&Mehrez09, 4.56] has also addressed the pros and cons involved with tree based routing architectures and island-style and have proposed a mix of both to achieve better results compared to island-style-only architecture.

Automatic Layout Generation of FPGAs

Efficient silicon implementation has a huge impact on the characteristics of FPGA. State of the art FPGAs are fully/partially custom designed and spend huge effort on layouts to get optimal implementation of the architecture. Works like [Toronto-Giles, 4.13][Mrabet09, 4.57][Parvez10, 4.53] have proposed methods and tools to facilitate automatic generation of FPGA layouts that are not optimal like real state of art layouts in terms of density etc. but can provide a good tradeoff in some cases to rapidly create layouts of FPGA architecture on different technology nodes saving long manual layout development/migration time.

2.1.4 Beyond Classics emerging works

So far this chapter has discussed several aspects and challenges concerning FPGAs architecture. It is worth noticing that with all the innovations and research done to improve FPGAs the fundamentals have remained quite same since beginning ([Xilinx_Freeman89, 3.1]). FPGAs are composed of programmable LUTs, programmable interconnect connects these LUTs. SRAM is still the de-facto configuration element for most of FPGAs (Xilinx+Altera SRAM FPGAs alone have above 80% market share [Coudert09, 2.14]). During last few years research has also been conducted in some non conventional or beyond classics directions for addressing the challenges of FPGAs. Most of them are still more at research level and will take a while for maturity before they may reach mainstream in industry. Their potentials to address specific FPGA challenges are very promising. This section presents a brief overview of some of the trends and how they are interesting for FPGA architecture challenges. For simplicity both academic and industrial solutions are presented together. MRAMs will be discussed in a bit more detail in this thesis work as they are a complimentary contribution of this thesis to observe, describe and present the industrial potential of MRAMs for FPGAs in general and in particular to eFPGAs (chapter 5.4 will address them in more detail).

The characterization can be generalized into two main categories, first innovations in architectures using un-conventional techniques, and second innovations for using un-conventional technologies for configuration of FPGAs. They are briefly highlighted below.

Architecture

The FPGAs are well known for huge cost of routing interconnect and high power consumption. With Moore's law the density and die size of FPGAs have grown. For very large multi-million gates FPGAs, the clock distribution is also becoming a great challenge due to large die size. Several modern techniques used in SoCs (GALS, Mesochoronous, Asynchronous etc.) are relatively difficult to exploit on FPGAs due to the inherent issues of FPGAs fundamental architecture. [Kuon.Tessier.Rose08, 4.2] has discussed some un-conventional research directions to address some challenges of FPGAs. Among some notable directions are asynchronous FPGAs that help to have high speed operation, reduce power and ease clock issues. Achronix (see in next section and patents appendix A1) is a commercial example of this direction. Academic research is also conducted in this direction [Teifel&Manohar, 4.59-4.62][Chaudhuri09, 4.102]. Another notable commercial example is Tabula (see in next section and patents appendix A1), it has created a highly time multiplexed FPGA-like programmable device which makes specialized use of dynamic reconfiguration at a very high speed. Network on Chip (NoC) have also gained attention as a good candidate to address the routing and ease of clock distribution kind of challenges of FPGAs. FPGAs with their large heterogeneous mesh of programmable and hard elements inherently provide a complex Systom on Chip (SoC) like scenario where potentials of NoCs can be fully/partially exploited to observe what benefits can be obtained. There are several research issues to address and will take a while to judge how suitable NoCs/Hybrid NoC+Classical routing can be for mainstream FPGAs. Concepts to use NoC (packet switching), time-multiplexed routing etc., are under investigation among the FPGA vendors (see patents appendix A1 for more details) [Xilinx_Young09, 3.30][Xilinx_Trimberger07, 3.29][M2000_Reblewski09, 3.75]. Another recent research idea from academics is using high speed serial communication (packets) in routing compared to immense parallel network of routing tracks (big portion is always unused for any application mapped) to have a more efficient architecture [Teehan&Lemieux09, 4.17]. Work on 3D technologies such as die stacking etc. is also under investigation and now commercially begins on 28nm Xilinx FPGAs [Xilinx10; 1.9, 1.10]. In longer term scientists are investigating about nanotechnologies if they can replace or compliment CMOS in FPGAs [PanelFPGA09, 2.10].

Configuration

Besides the architecture a significant portion of FPGA in terms of Area and Power consumption is the configuration of FPGAs that makes the FPGA function the mapped application. SRAM is still the king in this regard. Flash is the second most famous memory used in FPGAs. Research has been conducted to find new memories and also new ways to put that memory in the architecture to obtain benefits other than just classical ones (volatile/non-volatile, area etc.). [Chen.Lewis.Mitra09, 4.107] as a joint work of academics (Stanford) and industry (Altera) have shown the early concept of using nano-electro-mechanical (NEM) relays for use in FPGAs routing. Their zero leakage and low resistance is very attractive for power reduction; they can be fabricated on top of standard CMOS giving fabrication and enhanced area reduction benefits. The hysteresis characteristics of NEM can also provide potentials of non-volatility. [Muller&Thakkar08, 4.108] from Berkeley also addresses a similar direction. [M2000_Ebeling10, 3.77] presents potentials of MEMS (micro-electro-mechanical-systems) for switching elements in FPGAs.

MRAMs (magneto-resistive RAMs) have also been seen as having great potentials for FPGAs in terms of non-volatility, dynamic reconfiguration, multi-context, fabrication ease with conventional CMOS etc. *Chapter 5.4 will address them in more detail*. The research work/references regarding MRAMs presented in this thesis is from University of Montpellier LIRMM lab and its close collaboration with Menta, recognized by industry observers [EEtimes09, 4.78][EEtimes10, 4.79]. A comprehensive overview of potentials can be found in [S-4][S-5]. Deeper details are beyond scope of discussions and can be found in [Bruchon07, 4.73][Guillemenet&Torres08, 4.74][Guillemenet&Torres08, 4.75][Guillemenet&Torres10, 4.76][Cargnini10, 4.77][LIRMM, 4.80].

2.2 Systems with embedded FPGAs (eFPGAs)

Since this thesis work is focused on FPGA-like embedded FPGAs, section 2.1 covered extensively the fundamentals of FPGAs as knowing them is essential to conduct research on eFPGA architecture. This section in a similar fashion like section 2.1 investigates embedded FPGAs (eFPGAs) or similar solutions in industry and academics for research motivations. Figure 2.28 illustrates a theme diagram highlighting the fundamental concept of eFPGA in systems. The present day chips are composed of plenty of IPs (Intellectual Property) like microprocessors, memories, special functions etc. eFPGA IPs can help providing programmable IPs which brings the well known benefits of FPGAs directly inside the SoC. Chapter 5 will address this topic in detail.

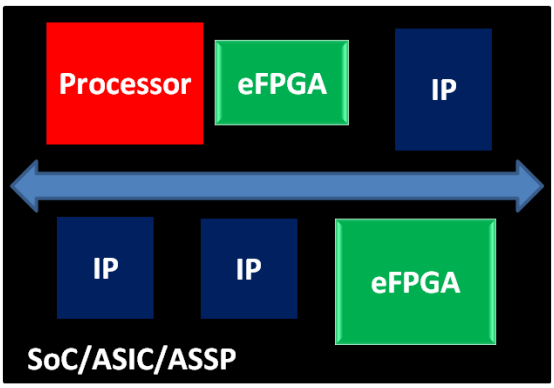


Fig. 2.28: Theme concept of eFPGA IP in SoC/ASIC/ASSP

2.2.1 Industry

The concept of eFPGAs is not new in industry; however it never had any wide success. That remained a big challenge for this thesis work and in addition for the thesis being in an industrial context was an added motivation for deeply analyzing industry, state of art and past researches not only for scientific inspirations but also focusing on why they failed and learn from those mistakes (section 2.3 will further address these issues in general industrial survey). This section discusses some examples from industry that exist or existed representing the eFPGA/eFPGA-like concept.

1- General overview

Some companies that provide/provided eFPGAs or systems with eFPGAs are discussed below.

Abound Logic (former M2000)

Abound Logic (former M2000) is the most notable example of company that provided embedded FPGAs (FlexEOS macros) based on ST process [D&R04, 2.23][D&R05, 2.24][D&R05, 2.25][D&R05, 2.26][EETimes06_C.Gross, 2.29]. The products that are publically known to have used their eFPGAs are the 130nm GreenField-STW21000 [EETimes07_P.Clarke, 2.30][ST05, 2.27] microcontrollers of ST for wireless market and the Morpheus project chip [2.28]. Figure 2.29 shows the 90nm Morpheus project chip block diagram and die picture. A 12 mm² eFPGA of M2000 is highlighted in the figure. The FlexEOS SRAM based eFPGAs of M2000 had logic densities of 1340LUT4 per mm² on 90nm. However from the website of company it seems they have abandoned the eFPGAs business, now they make low cost ultra high density TSMC 65nm FPGAs. Unfortunately there is very limited public information available about the device architecture, tool and tools efficiencies, customers and researchers feedback of experiments done on those eFPGAs to get some scientific observation etc. for their abandoned FlexEOS eFPGAs. Analyzing some of their patents [Reblewski&Lepape03, 3.72][Lepape09, 3.73][Lepape10, 3.74] it appears that their architecture is hierarchical/partly hierarchical (not classical island-style) that leads to their superior logic densities competitive differentiation claim against Xilinx & Altera.

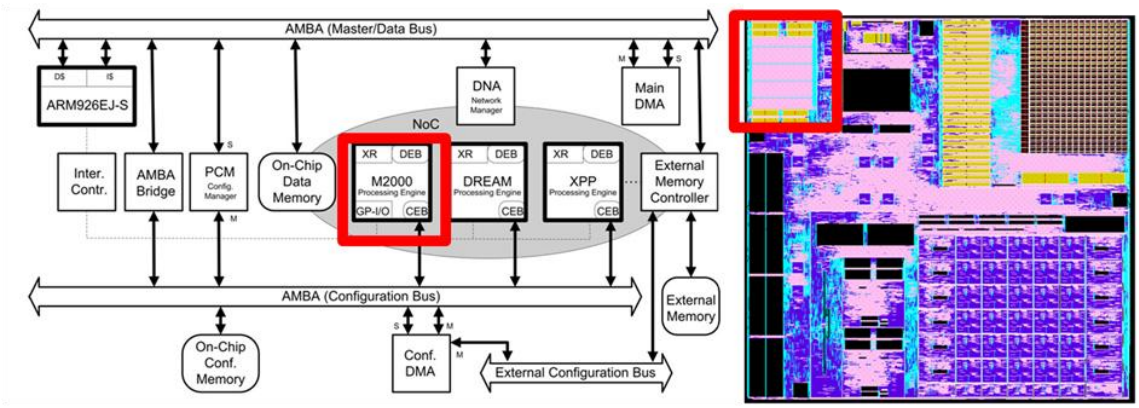


Fig. 2.29: 90nm Morpheus Chip with eFPGA of Abound Logic/M2000 [2.28]

Atmel (CAP MCUs)

The CAP microcontrollers of Atmel [2.34] are an interesting commercial example of importance of eFPGAs and particular challenges related with eFPGAs when chip designers consider their use. The CAP MCU provides MCU infrastructure based on ARM7 processor and AMBA bus. It has a metal programmable (MP) block efficiently connected with AMBA where customized functions can be implemented to have a differentiated product. The MP block is only one time programmable (in Fab in few weeks) and has a gate array of high density close to standard cells (not FPGA-like structured ASIC architecture). This allows very compact, faster and power efficient implementations but only one time programmable (in many real cases it is a good proposition like Structured ASICs one time programmability vs reconfigurable silicon heavy solution). Figure 2.30 shows the concept of Atmel CAP. The differentiation functions are implemented in a separate FPGA (development board) for evaluations and then can be easily migrated to the MP block for mass production using tools of Atmel to create a single chip MCU solution.

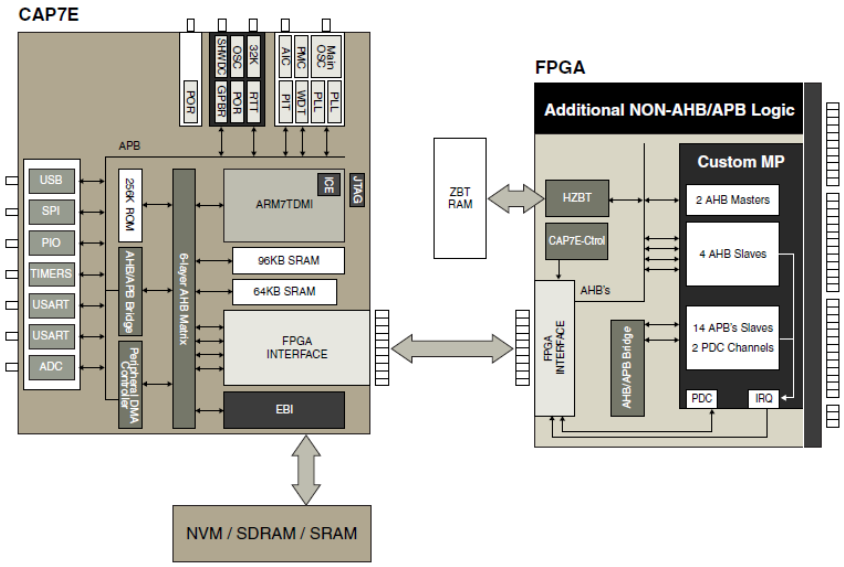


Fig. 2.30: Atmel CAP Microcontrollers with Metal Programmed (MP) embedded fabric [2.34]

IBM-Xilinx eFPGAs for SoCs (2002)

In 2002 IBM and Xilinx made a partnership to have embedded FPGAs of Xilinx on IBM 90nm process for the Cu-08 ASICs products of IBM. Three cores were planned from 10K to 40K gates with up to 640 I/Os. Multiple eFPGAs could also be integrated within a single ASIC device [Xilinx02, 2.35]. There is not much information available about the Cu-08 product [IBM, 2.36] and evidence of continued products in this direction. However around same time a similar thing happened in Xilinx FPGAs of the well known hard

IBM Power PC processor inclusion in the Virtex2 pro devices, that continued for several generations of Xilinx FPGAs.

Varicore (Actel)

In 2001 Actel introduced a new class of SRAM based Varicore embedded FPGA IP cores based on 180nm Chartered Semiconductor Manufacturing process. The cores were used in ASICs, ASSPs, SoC devices to help speed products to market and increase the life of those products once in the market. The cores were available from Actel for customers using Chartered process [D&R01, 2.37] [EEtimes01, 2.38] (fab dependent solution). The product line was abandoned by Actel and no technology information details about these cores can now be found on their web.

In a 2006 annual report [Actel06, 2.40] Actel explains some background and reason of abandoning the product as. "We introduced our VariCore embeddedable reprogrammable gate array (EPGA) logic core based on SRAM technology in 2001. Revenues from VariCore EPGAs did not materialize and the development of a more advanced VariCore EPGA was cancelled. In this case, a market that we believed would develop did not emerge". "In 2000, we acquired Prosys Technology, Inc. (Prosys) for consideration valued at $26.2 million. We acquired Prosys for technology used in our VariCore FPGA logic core, which was introduced in 2001 but for which no market emerged".

In addition in this report Actel describes their move to make SRAM based FPGAs in late 1990s that they abandoned. "In 1999, we acquired AGL for consideration valued at $7.2 million. We acquired AGL for technology used in the unsuccessful development of an SRAM-based FPGA".

Adaptive Silicon

Adaptive Silicon [D&R01, 2.41][D&R01, 2.42][EETimes99, 2.43] in 2001 proposed Multi-Scale Array (MSA) embedded programmable logic cores at 180nm from LSI Logic and TSMC. Unlike most others MSA was not LUT based but coarse grain architecture with array of 4-bit ALUs that could be programmed by their software tools called Millennium PLC using RTL flow. Due to being coarse grain the solution was more silicon efficient compared to fine grain LUTs. The company did not succeed and no technical information is available anymore except industry news/press releases.

Leopard Logic

In 2002 Leopard Logic proposed HyperBlox FP SRAM based embedded FPGA cores that were process specific to TSMC 180nm and also were part of TSMC's IP Alliance program [D&R02, 2.44]. Their tools suite Leopard Logic's ToolBlox allowed easy integration of these cores in SoCs. The company went out of business like others and no detailed information is now available about their architecture and tools except industry news/press releases.

2- FPGAs-2010 with Hard Processors (FPGA or eFPGA!)

The two opposite trends: FPGA vendors putting hard blocks in their devices and ASICs/ASSPs vendors looking for limited flexibility in their devices in form of eFPGAs or similar cores has always remained in discussions in industry. It was shown above that two leading FPGA vendors (Xilinx, Actel) also at a time proposed embedded FPGAs to specific client (Xilinx-IBM) or general market (Actel Varicore) but abandoned soon. Altera never went in this direction like Xilinx and Actel. The view point of opposite side were (now getting further relevant due to power) also very prominent that customers do not want full PLDs but only limited flexibility only where needed. The programmable logic inherently is much silicon expensive compared to equivalent hard implementation [EETimes99_LSI, 2.43]. It is well known that industry never saw a significant success with eFPGAs. The other side (FPGAs) continued to succeed over passage of time in putting hard blocks to continue making their devices a competitive solution against the

ASICs. As of 2010 FPGAs have reached to a point in heterogeneity that their fine grained programmable logic have started to seem like an eFPGA surrounded by several hard blocks and bus infrastructures like SoCs and in fact have become more like a Configurable SoC than FPGA [Xilinx10_P.Lysaght, 2.5][Altera10_V.Betz, 2.4][Xilinx, 1.2][Altera, 1.27][Actel, 1.28].

Figures 2.31 show the latest devices from Actel and Xilinx with Hard processor blocks of ARM. It can be observed that FPGAs are approaching SoCs, SoCs like design challenges, SoCs like programming. It is interesting to look that the design on modern FPGAs is becoming more and more IP centric with Processor, the surrounding IP infrastructure of that Processor with more and more hard blocks with fine gained classical structure shrinking continuously for differentiated functions (eFPGAs are meeting FPGAs on FPGAs!). The 28nm EPP of Xilinx is much like a SoC with eFPGA, coming from an FPGA vendor [Xilinx10_P.Lysaght, 2.5].

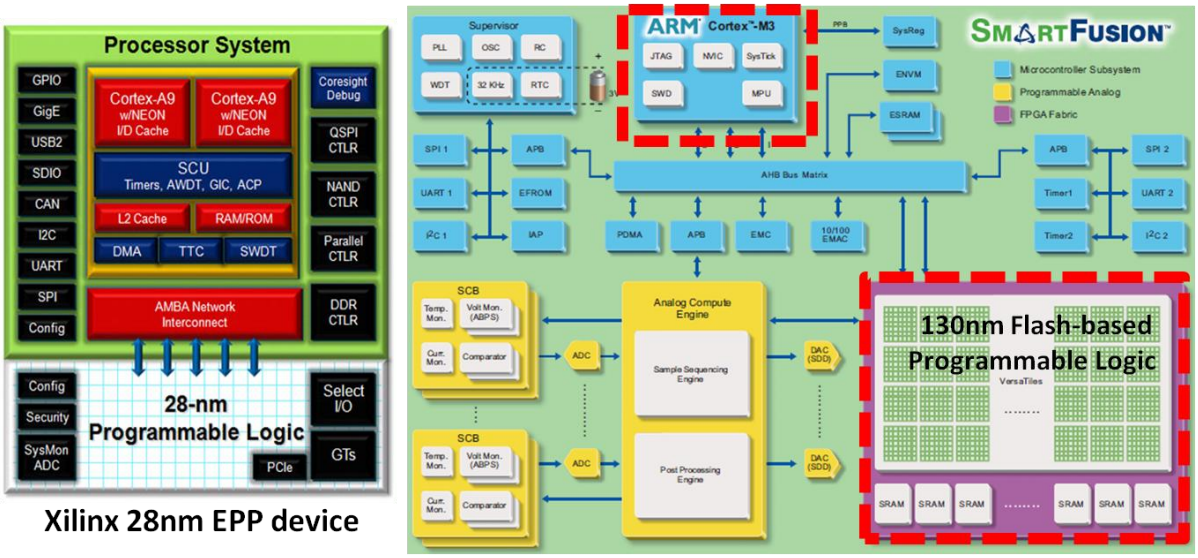


Fig. 2.31: FPGAs with Hard ARM: Xilinx 28nm EPP and Actel 130nm Smart Fusion (source: web)

2.2.2 Academics

The work done in academics in this regard can be divided into two main classes. One direction is using the classical FPGA like LUT based architecture and the other is coarse grain architectures which are widely addressed in academics. FPGA-like architecture will be discussed in detail as this thesis work is based on FPGA-like eFPGAs and briefly overview of the references of coarse-grain research will be provided.

1- FPGA-like embedded FPGAs

The previous section discussed about the two opposite side trends of industry. FPGAs seeking reducing their gap with ASICs with hard blocks and ASICs seeking flexibility using embedded programmable cores (did not succeed but nevertheless many attempts were made due to obvious attractive reasons and potentials that embedded programmable cores can bring to ASICs). Different works in academics have addressed issues related to eFPGAs and their system integration scenario.

In [Wilton.Kafafi.Saleh05, 4.31] authors have addressed the challenges and issues involved with eFPGAs, in particular soft eFPGAs. This work was done around 2003-2004 and the examples of failed businesses of several companies around 2001-2002 which were discussed above were an inspirational example for them. It addresses nicely the fundamental issues related with eFPGAs, they summarized their views as follows.

1) Tools for the design and integration of programmable fabrics are not widely available as yet. This is somewhat of a chicken-and-egg problem: existing tools and flows will not be enhanced to support the seamless integration of programmable logic cores until this design technique becomes mainstream, and the design

technique will not become mainstream until the tools are enhanced to support programmable logic cores. However, as chip design costs escalate, the economics of chip design will be a strong driver for increased hardware programmability.

2) Programmable logic cores come in relatively fixed formats. That is, the integrated circuit designer cannot modify the overall size of the fabric or the internal structure of the programmable logic core. The integrated circuit designer must choose a programmable logic core that is closest to the desired size; this could lead to wastage of chip area. This can be addressed by providing tiles of programmable logic that can be snapped together to form a design logic fabric of the desired size to minimize the area penalty.

3) Embedded programmable logic is not as efficient as hardwired logic in terms of area, power and speed. There are, however, special-purpose fabric generators emerging that can provide a better tradeoff between these specifications, depending on the target application.

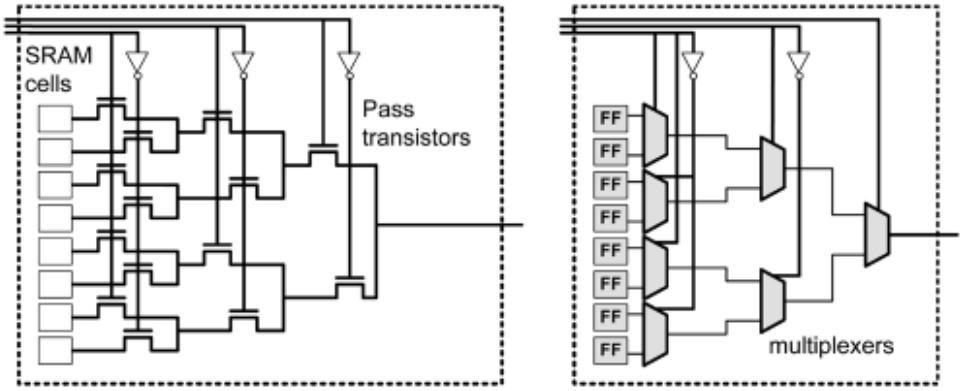


Fig. 2.32: Hardware comparison between standard custom FPGA (left) and Soft FPGA (right) [4.31]

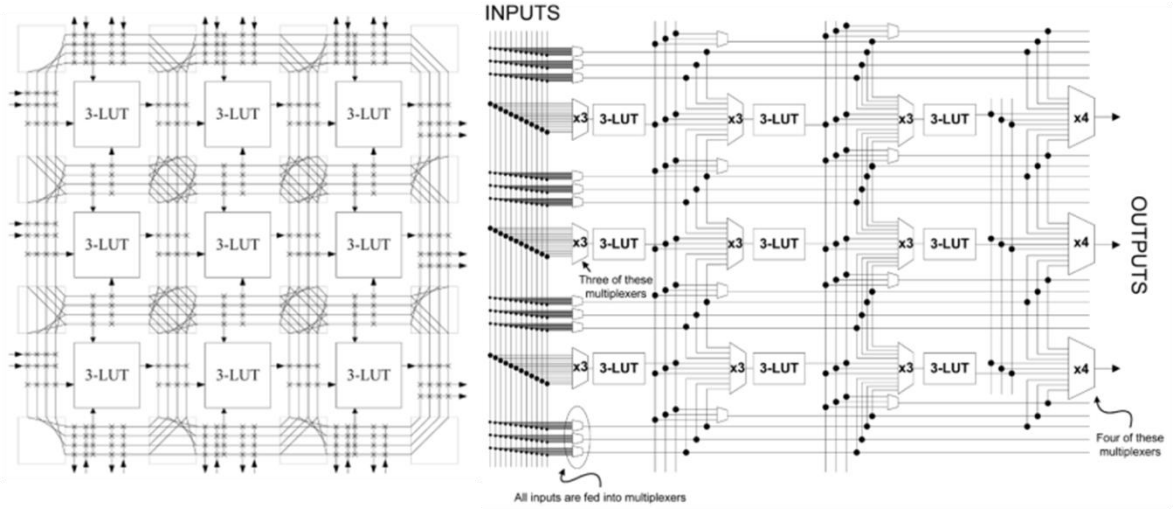


Fig. 2.33: Directional (left) and Gradual (right) [4.31]

To address some of these issues their work proposed soft eFPGAs which unlike custom FPGAs/eFPGAs are written in HDL and can be synthesized using standard ASIC tools making them easier to integrate in the SoCs. As they are soft and mapped on the standard cell library they use flip-flops and multiplexers of the standard cell library unlike pass-transistors, tristate-buffers, SRAM cells that are used in FPGAs or hard eFPGA blocks. Figure 2.32 shows this difference. They proposed two architectures for creating soft eFPGAs shown in figure 2.33. First a directional architecture that is similar to the classical island-style architecture but only allows data flow in one direction to avoid combinational loops for the synthesis tool while integrating it in SoC. Moreover it only implements combinational circuits (flip-flops are used for configuration but no flip-flop associated with LUT for sequential logic). Second a gradual architecture that attempts to further improve area efficiency. This research work focused very small sized eFPGAs (less than 100 LUTs) where the architecture is simplified to implement small combinational logics. VPR was modified to map applications on these proposed architectures.

[Akenova.Lemieux.Saleh07, 4.14] has addressed reducing the gap between the soft eFPGAs compared to hard eFPGAs with their proposed Soft++ methodology. As the FPGA architecture is highly regular and composed of fundamentally two elements (configuration cells and multiplexers) other than flip-flops in LUT-FF pair for sequential implementation. This work has proposed to create standard cells of custom layout of basic elements like SRAM and multiplexers, and use these cells (they call them tactical cells) instead of general standard cells of full CMOS multiplexers and flip-flops for configuration element. This allows greatly improving the silicon efficiency of soft eFPGAs while ease of using standard synthesis flows, hence providing a semi-custom solution between pure soft and full custom hard eFPGA.

[Quinton&Wilton07, 4.27] addresses the challenges regarding interfacing eFPGAs with buses and propose solutions in this regard.

[Neumann&Noll08, 4.87] presents a framework to create customizable eFPGAs from higher level of description of architecture parameters for arithmetic oriented application domains.

2- Coarse-Grain Reconfigurable Architectures

Coarse grain architectures perform computation at a level higher than 1-bit fine grain LUT computation which is standard among FPGAs (there are several coarse grain blocks in modern FPGAs but they are connected with fine grain computation model of LUTs and interconnect logic). Compared to research on FPGAs or eFPGAs, the area of coarse grain architecture is widely and deeply addressed in academics. There are several directions in which research is conducted on coarse grain, e.g. Array of ALUs, custom processors with reconfigurable data paths, array of very small RISC machines etc. On industrial side unfortunately the history of coarse grain is painful with several failed attempts from many companies (next section will address them/issues with them).

Deeper details and survey on research done on coarse-grain architectures (like was done for FPGAs) is beyond the scope of this thesis. However general overview of these works was done for inspiration. A good comprehensive survey of research done in this area can be found in [Hartenstein01, 4.64][Brunelli08, 4.89]. Some notable examples include Totem Project [4.97][Huck&Compton, 4.98][Compton03, 4.99], XiRisc [Lodi03, 4.104], GARP [Hauser&Wawrzynek97, 4.105], PipeRench [Goldstein00, 4.106], Xputer [Becker97, 4.90] etc. [Becker&Vorbach03, 4.91] presented a system of coarse grain hardware acceleration unit (PactXPP) connected with general purpose LEON processor with AMBA bus.

A detailed overview, potentials of reconfigurable computing, challenges and efforts for education in this direction, the potential flaws of current Von-Neumann based computer science can be best achieved from keynote talks and related publications of Prof. Reiner Hartenstein [4.65-4.72].

Coarse-Grain vs Multicore: Coarse-Grain research remained quite hot throughout the 1990s and early 2000s (above mentioned references can provide an overview) in academia, however it failed to succeed/penetrate in industry (FPGAs dominated, next section will provide overview), which lowered the momentum of research in this direction. While it still is partly active in research but the rise of another wave of multicore era in second half of 2000s has further dampened the research in coarse-grain direction as majority of researchers switched to this new wave which partly uses inspirations of coarse-grain also (computation in space). At present from research and industrial stand point the major competitive waves in Reconfigurable Computing domain are FPGAs and rising multicore. Next section will provide further details along with several other industrial trends, failure reasons of past etc.

2.3 The Semiconductor Industry

This chapter is explaining the contribution R of the thesis work. Chapter 1 explained the nature and dimensions of the contribution of this work. The above two sections 2.1 and 2.2 presented the details of state of the art in industry and academic research for the areas of FPGAs and embedded FPGAs. This section outlines a general industrial survey with taking into account commercial aspects also, since the discussions will be incomplete without realizing this aspect of industry as no real-life challenge can be met without knowing real-life (CIFRE motivation [5.11]). Nevertheless to maintain the scientific intent the thesis has restricted the discussions of commercial investigations to their general scientific conclusions for technological challenges that are interesting and essential to know for all research community. The general overview of the contents in this section is as follows. Sections 2.3.1 briefly outlines the investigations of this thesis work done to better understand industrial aspects. The rest of the sections will detail the comprehensive overview of major findings of the research done for survey of types and potentials of several programmable technologies in industry [S-3a, S-3b] to get global overview of potentials and challenges regarding eFPGAs.

2.3.1 Understanding Industry in general

One of the major objectives of this work was to get trained for industry [5.11]. So understanding the basics of complexities of the Industry (technical and commercial) to well prepare to be integrated in industry was an important objective. Efforts, studies and several contributions [S-1][S-2][S-3][S-6][S-7] were made in this regard. As commercial discussions are kept to minimum, below are two important areas that helped a lot for getting educated in this regard.

1- Industrial articles and press news (From IEEEtimes to EEtimes)

There is a quite known phenomenon in industry regarding eFPGAs, "*everyone who tried to do this died or left the business*". Since this thesis work was in an industrial context, finding/discovering answers to such issues was almost mandatory. Any researcher who finds out that sometimes strange things happen in reality (this thesis work was no different) he/she obviously goes in a deep *curiosity of Why?*. Things like how can this thing ever fail?, Why NoCs have not replaced buses?, Why industry is not going crazy for nano-technologies?, Why bio-inspiration rarely inspired anyone?, Why coarse-grain did not replace FPGAs?, FPGAs are so powerful and amazing why don't they replace everything?, … (hypothetical list can go on and on). To investigate such complex issues help was sought by deep study of industrial articles, press news, industrial events/conferences, webcasts etc. to make practical industrial survey. Key findings and conclusions of this commercial research will be outlined in next sections [S-3]. [5.7][5.8][5.9] also provided general understanding of industry.

2- Patents study of State of the Art

A general overview of state of the art technologies in most cases can be achieved by product brochures, datasheets, webinars, conferences etc. However to understand deeply the architectural innovations, patents are a good scientific literature (although their intent is not just scientific!). They comprehensively present highly innovative solutions and give an overview of how state of art manage/solve complicated *real-life* problems and provide good motivations and directions for research challenges. Dozens of patents of leading FPGA vendors were analyzed in the thesis work; some prominent ones are referred in [3].

Appendix A1 provides a comprehensive survey of *patents* study done for FPGA vendors focused on several aspects of FPGAs (logic block, routing architecture, tile-based layout, configuration, hard blocks, emerging trends etc.).

2.3.2 Global overview of Industry

This section provides a global overview of industry, the cyclic nature of industry and some major challenges it is currently facing.

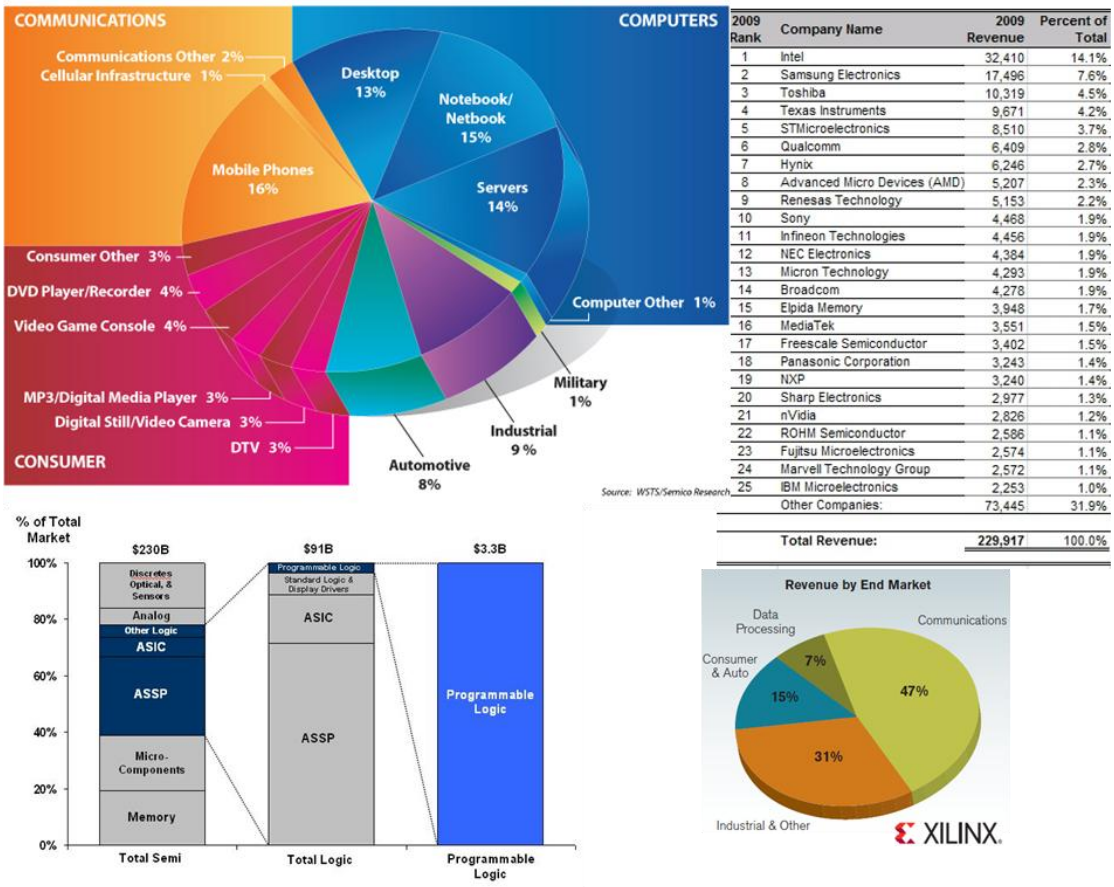
1- Semiconductor Industry markets and market leaders

Figure 2.34 provides overview of semiconductor industry in a nut shell for 2009. In 2009 global semiconductor revenues were around 230 billion dollars [2.1]. On the top left a detailed pie graph [EEtimes10, 2.15] shows distribution of major markets. The dominance of Computers and Communications is prominent. On the right there is a table of top 25 semiconductor companies [2.1] which represent almost 70% of entire semiconductor industry. Further among them high dominance of Intel and several memory giant companies is prominent (obvious from the huge share of computers in the industry).

On the bottom left [S-3b] the bar graph provides the details of market share from technology point of view (compared to market type in top left graph) with special emphasis on programmable logic devices market. It is clearly evident that ASICs and in particular ASSPs have a very high market share compared to PLDs (FPGAs, CPLDs etc.). FPGAs with 3.3 billion dollars market roughly represent around 1.5% of 230 billion dollar semiconductor industry. This also gives a brief idea of the vastness and huge size of the overall industry. In the PLD market Xilinx and Altera are the top market leaders having more than 80% market share of the PLD market. Figure 2.35 shows the top FPGA vendors and their market share details and the overall FPGA market revenue change in last ten years (the 2001 dotcom bust and 2008-2009 economic crises can be immediately seen in that) [Coudert09, 2.14][EEtimes10, 2.20]. To get some insight of, in what kind of markets FPGAs are used a hint can be obtained from Xilinx (being no. 1 and Altera being direct competitor has similar markets too). On the bottom right of figure 2.34 is the graph of Xilinx revenue distributions based on market in 2010 (source: Xilinx website). The strongest segment evident is communications. Comparing this graph with the top left graph of figure 2.34 clearly shows in what kind of segments FPGAs go and how big those segments are in overall scenario. This on one hand shows still great room of new opportunities for FPGAs to grow in, and also shows why they are still a very small niche in industry as all the high and big markets are highly dominated by ASSPs, Computers etc. from top most semi conductor companies.

2- Makimoto's Wave: The cyclic nature of Industry

The iconic Moore's law [2.2] is the main driving source of industry for more than four decades (and still expected for around 1 more). During late 1980s Tsugio Makimoto in Hitachi observed another phenomenon regarding the cyclic nature of industry. He observed that for last several decades industry was switching between Standardization and Customization and that the duration after it switches is almost 10 year. He plotted a cyclic graph to show this phenomenon that became highly famous as Makimoto's wave in the 1990s and particularly for his forecast of boom of Field Programmable Devices [S-3b]. The wave is shown in figure 2.36. Surprisingly it still seems valid and as of now (2010) according to the graph the industry entered again in the 10 year customization cycle in around 2007. If one observe it seems quite valid as more customized solutions are getting prominent in industry, FPGAs top vendors also are providing specialized target domain specific FPGAs [S-3b]. [Xilinx07_Trimberger, 2.6] illustrates current period as the specialization age of FPGAs for creating more target optimized FPGAs.



2009 Rank	Company Name	2009 Revenue	Percent of Total
1	Intel	32,410	14.1%
2	Samsung Electronics	17,496	7.6%
3	Toshiba	10,319	4.5%
4	Texas Instruments	9,671	4.2%
5	STMicroelectronics	8,510	3.7%
6	Qualcomm	6,409	2.8%
7	Hynix	6,246	2.7%
8	Advanced Micro Devices (AMD)	5,207	2.3%
9	Renesas Technology	5,153	2.2%
10	Sony	4,468	1.9%
11	Infineon Technologies	4,456	1.9%
12	NEC Electronics	4,384	1.9%
13	Micron Technology	4,293	1.9%
14	Broadcom	4,278	1.9%
15	Elpida Memory	3,948	1.7%
16	MediaTek	3,551	1.5%
17	Freescale Semiconductor	3,402	1.5%
18	Panasonic Corporation	3,243	1.4%
19	NXP	3,240	1.4%
20	Sharp Electronics	2,977	1.3%
21	nVidia	2,826	1.2%
22	ROHM Semiconductor	2,586	1.1%
23	Fujitsu Microelectronics	2,574	1.1%
24	Marvell Technology Group	2,572	1.1%
25	IBM Microelectronics	2,253	1.0%
	Other Companies:	73,445	31.9%
	Total Revenue:	229,917	100.0%

Fig. 2.34: Semiconductor industry 2009 (230 B$) markets and market leaders [S-3b][2.1][2.15]

Worldwide FPGA/PLD vendor revenues and rankings, 2007-2008

Rank 2007	Rank 2008	Company	Revenue ($M) 2007	Revenue ($M) 2008	Revenue Change 2007-2008	Market Share 2008
1	1	Xilinx	1,809	1.906	5.4%	51.2%
2	2	Altera	1,216	1,323	8.8%	35.5%
3	3	Lattice Semiconductor	229	222	–3.1%	6.0
4	4	Actel	196	218	11.2%	5.9%
6	5	QuickLogic	28	23	–17.9%	0.6%
5	6	Cypress Semiconductor	32	21	–34.4%	0.6%
7	7	Atmel	14	9	–35.7%	0.2%
8	8	Chengdu Sino Microelectronics System	4	3	–25.0%	0.1%
		Others	0	0	NM	0.0%
		Total Market	**3,528**	**3,725**	**5.6%**	**100.0%**

Source: Gartner

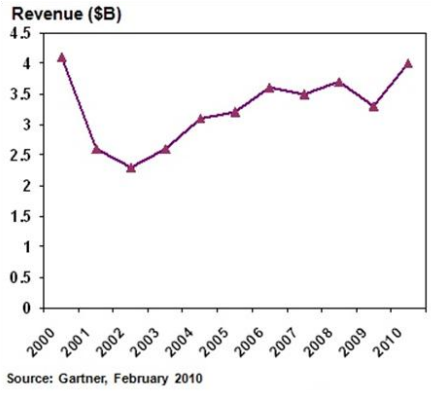
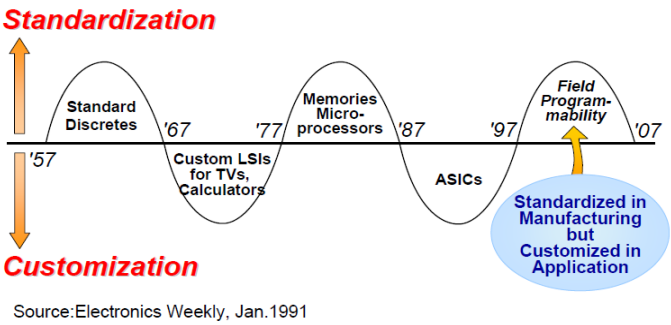


Fig. 2.35: Top FPGA/PLD vendors market share and revenues [2.14][2.20]



Source:Electronics Weekly, Jan.1991

Fig. 2.36: Makimoto's Wave: Cyclic nature of Industry (Source: open web)

3- Challenges of Power Consumption

In general at present, entire semiconductor industry is under great challenge for power consumption. As Moore's law crossed 90nm, the challenges of power, in particular static power became very prominent throughout industry. The birth of MPSoCs around mid 2000s was also result of that. The issues of power challenges are multifold. Firstly the biggest source of reducing dynamic power historically was scaling of the supply voltage. As technologies have moved beyond 90nm the supply voltage has almost flattened around 1.0V, furthermore in these finer geometries static power has become significant due to leakage (section 2.1.2 presented the scenario for FPGAs). On the products side trends towards more and more portable devices with high computational power is mainstream, making obvious the challenges of power concerning battery based devices. Further on top the rising costs of energy and green requirements are further pushing demands for non battery based devices to be also power efficient. The issues regarding power consumption of data centers and servers are further pushing industry to innovate for low power devices. Figure 2.34 presented huge market shares of ASSPs/ASICs compared to FPGAs, power is one of the major factors behind that, as despite of the numerous benefits of FPGAs, the large silicon gap that they have compared to ASSPs/ASICs makes them difficult to enter in several high volume applications.

2.3.3- Types of Programmable Hardware

This section briefly outlines different classes of hardware that exist in industry and can come under the category of programmable hardware.

1- High-end MPSoCs

On the highest end of industry are the general purpose processors from Intel, AMD etc. They represent the historic computers evolution and its dominance in industry, directly and indirectly (companion markets which also boom because of it like memories etc.). In mid 2000s when issues of power became imminent the trend of Multicore was set in industry by these high end products, symbolizing the challenges of power consumption. Figure 2.34 presented the high market share of computers in industry. This is a very specialized class of products with very limited competition from other technologies. This category is beyond scope of discussions of this work.

2- Heterogeneous MPSoC Platforms

Coming one step down, another very prominent class are heterogeneous MPSoC platforms that are getting increasingly important in industry due to rise in low power portable devices. Micro Controller Units (MCUs) can also be placed in this category. The discussions will focus more on larger systems for ease of discussion as they cover some of the main features of MCUs also. Such devices are rich in IPs with a processor centric design that controls several IPs as specialized accelerators. ARM is very dominant in this market as the processor IP provider. Figure 2.37 shows the well known OMAP4 platform of TI which presents the concept; a dual core ARM processor is surrounded by numerous IPs (also DSPs) to perform specialized functionalities to augment the Processor. Similar competitive devices exists from vendors like Samsung, Qualcomm, Broadcom, Nvidia (tegra) and most recent well known A4 chip from Apple. Such devices are mainly targeted for mobile devices, providing a platform with high computational power at low power. Figure 2.34 showed mobile devices are becoming a very prominent market in industry and so is among a major focus of several large players (including Intel with Atom and AMD upcoming Bobcat processors). In such devices the main source of flexibility and differentiation for the customer is through the processor by software (iPhone is very prominent example of software differentiation).

3- FPGAs

FPGAs are most prominent devices regarding programmable hardware. They provide the huge flexibility to change or upgrade the design after fabrication, saving huge costs that will need to be spent if using ASSPs/ASICs. However as was discussed in previous sections, the large silicon gap of FPGAs prevent

them to enter in high end markets that are highly dominated by ASSPs and computers (the two categories discussed above), figure 2.38 illustrates that.

However with Moore's law, getting ever more logic densities and continued architectural innovations FPGAs are getting increasingly popular in several medium or lower volume applications where their ease due to flexibility provides a good tradeoff for the high cost needed to create ASICs/ASSPs on the newest nodes (FPGA leaders are always among the first adopters of newest node to get competitive benefits, with 28nm devices coming soon). The newer trends of processor centric FPGAs from Xilinx shown in figure 2.39 will further push FPGAs to enter in non conventional markets.

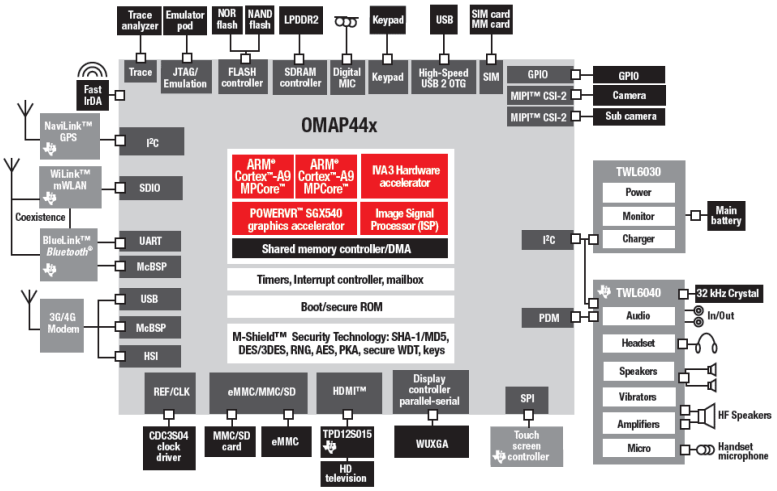


Fig. 2.37: OMAP4 platform of TI (source: TI web)

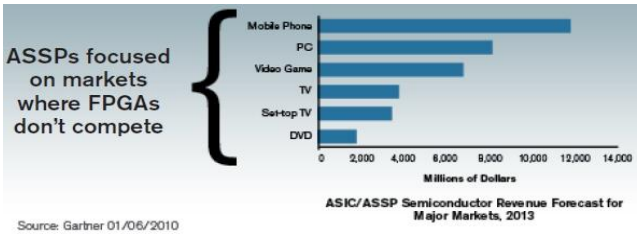


Fig. 2.38: Main markets for ASSPs (source: Xilinx web)

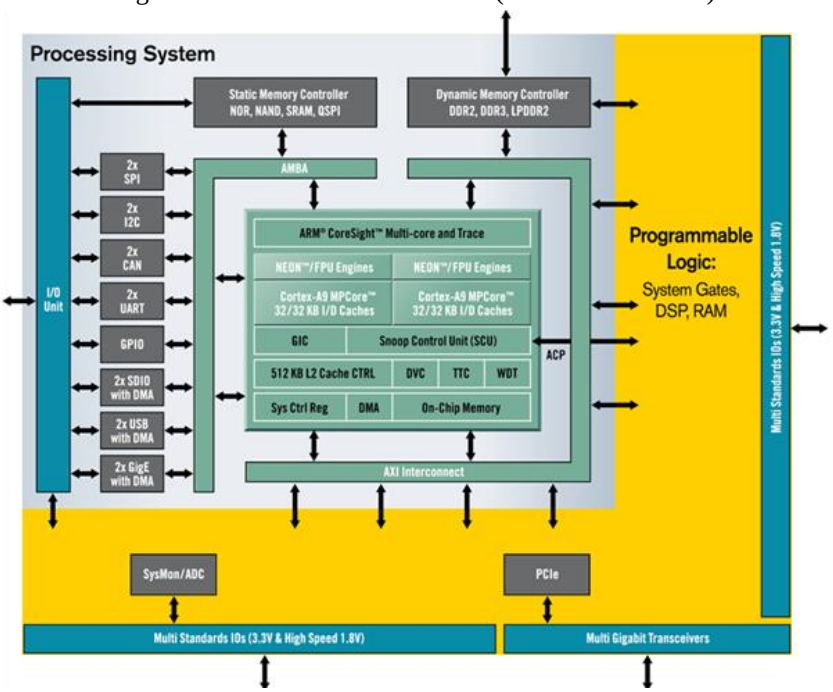


Fig. 2.39: Xilinx 28nm Zynq-EPP devices (source: Xilinx web), {Note: source image is blurry!}

4- Versus (vs) FPGAs (MPPAs, Coarse-Grain, Structured ASICs etc.)

As was discussed above, FPGAs are most prominent programmable devices but have a large silicon gap in terms of area, power and speed. There are several other dimensions (their market share is very small compared to FPGAs) to address this issue to create programmable solutions that have flexibility similar to FPGAs but are more silicon efficient. They can be characterized into three major categories, Massively Parallel Processor Arrays (MPPAs), Coarse-Grain and Structured ASICs.

MPPAs are a relatively new type of devices consisting of an array of processors. They gained traction after the industry moved to Multicore due to power issues. The central theme of MPPAs is to use processor of relatively small size in large quantity to perform FPGA-like (non Von Neumann) computing. The efficient programming of these devices is still relatively complex due to parallel programming challenges, setting aside that issue they have easier software programming model compared to RTL programming of FPGAs which is much faster from compilation point of view that is getting significant in FPGAs as their densities have immensely increased with Moore's law. Furthermore due to coarser grain nature they have good potentials to exploit features like DVFS (Dynamic Voltage and Frequency Scaling), adaptability etc. that are relatively complicated and difficult on FPGAs due to their inherent fine grain nature of hardware. Figure 2.40 on the left hand side shows an example of MPPA architecture from Tilera (tilera.com). More details about MPPAs can be found in [Butts08, 2.11].

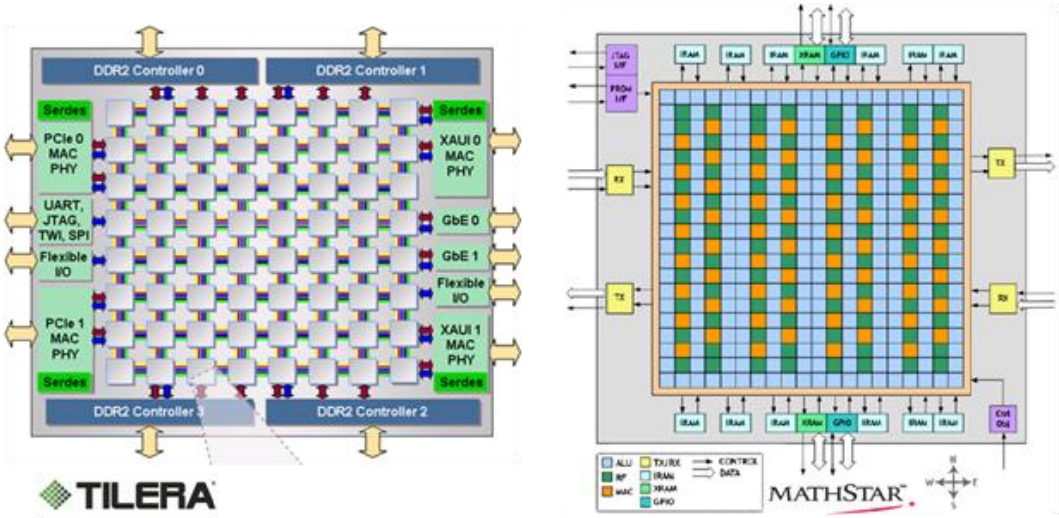


Fig. 2.40: MPPAs and Coarse Grain architecture styles

Coarse Grain architectures exist in multiple forms; it is not simple to characterize them like FPGAs and MPPAs. In general, the coarse grain architecture in contrast to MPPAs use smaller computational blocks, like ALUs, multipliers etc. to perform the computation. As the computation is block level compared to 1-bit fine grain modal of FPGAs, the solutions are more silicon efficient. The programming of coarse grain is mostly compile-based (software) like MPPAs. The right hand side of figure 2.40 shows an example of coarse grain architecture from Mathstar (source: web) called FPOA (Field Programmable Object Array).

Structured ASICs are a bit different class of devices; they are programmable but one time only. They are based on FPGA architecture where the programming of configuration is done at mask-level thereby significantly reducing the silicon overhead due to elimination of large amount of configuration SRAM cells in the device hence significantly improves the silicon gap of FPGAs from ASICs. They are offered by leading FPGA vendors (Xilinx EasyPath, Altera HardCopy) and also from some other companies, a prominent example is eASIC with their nextreme devices (easic.com).

2.3.4- FPGAs and vs FPGAs

Section 2.3.3 covered in general different types of programmable hardware that exist in industry in the order of their market share. As this thesis work is focused on FPGAs, this section investigates in a bit more detail the last two types of that section to have more insight about the pros and cons of them and why FPGAs are so dominant.

1- History of PLD startups

Figure 2.41 shows a detailed history graph of several companies who entered into programmable business for last almost 3 decades. It also shows the business status of the companies, the throughout success of current top 4 FPGA vendors (Xilinx, Altera, Lattice, Actel) is prominent in it. It is interesting to observe so many have tried to enter into PLD business (including very big names like Intel, IBM, Samsung, Toshiba etc.) and failed or abandoned the business.

2- Dominance of FPGAs: Fundamental Pros & Cons

This section briefly addresses the fundamental dominating potentials of FPGAs from technology stand point (setting aside the business duopoly of the Big-2) and some fundamental cons that keep them still small niche compared to ASSPs.

RTL Programming

Since their invention in mid 1980s the fundamental target and objective of FPGAs has remained the field programmable capability of prototyping the ASICs. Based on that the model of programming FPGAs is inherently HDL. It is the foundation source of designing the ASICs, is highly mature and industry standard with state of the art tools support.

Universal Nature/Prototype Power

The strongest strength of FPGAs is that they have universal capabilities due to their prototype ability and HDL programming model. This is not fully true for a microprocessor in a sense that even microprocessor can be prototyped on FPGA and soft processors are mainstream, the inverse is quite complex (although theoretically possible but too slow, well known simulation vs prototype speed issues). This power helps FPGAs absorb complex functionalities in form of Hard Macro blocks. It can be a processor, an IP or anything else. Since the programming model is HDL it gives instant usability of the component without any burden of new standards or languages. Highly mature in-house or 3rd party synthesis tools are available due to standard RTL flow.

IP eco-system leverage

The RTL flow of FPGAs provides an added benefit to IP eco-system of the industry. It is easier to port IPs both for ASICs and FPGAs as both use RTL. This also holds true for FPGAs of different vendors because they all use RTL flow, so porting the design to another FPGA is not extremely complicated like it sometimes is in microprocessors where legacy code plays a high role in its success and market dominance. Furthermore as RTL is inherently parallel, mapped application is automatically optimally parallelized by CAD (Computer Aided Design) tools utilizing the best of the target hardware resources (this still is one of major difficulty for multicore/multicore-like solutions).

FPGAs have become Programmable Platform

Moore's law is the major driving source for all the industry and is particularly important for giant semiconductor companies, FPGAs success has also been highly attributed to it. Like Intel for microprocessors, the FPGA vendors are also always among the first adopter on new technology node and remain ahead in this regard to most of the remaining segments of industry, the regular nature of FPGA

architecture provides them added scaling edge. Both de facto FPGA giants (Xilinx and Altera) are hitting 28nm in 2011. By continuously following Moore's law and architectural upgrades based on changing market needs FPGAs have now become capable of implementing entire SoCs. In fact they have turned now in a complex heterogeneous mix of coarse-grain elements and classical fine grained LUTs. *The term FPGA (Field Programmable Gate Arrays) no longer correctly justify the capabilities of modern FPGAs*. As an example Xilinx has already started the term Programmable Platforms for its devices.

FPGAs vs ASICs Silicon Gap

On the down side, the high programmable and prototype capabilities come with a very high price in terms of Area, Power and Speed which makes FPGAs in many cases too hard to be used as a product. It was seen in previous section (2.1.3) that FPGAs have a gap of around 30-40X in terms of Area, 12-14X in terms of Power and 4-5X in Speed when compared to an ASIC [Kuon&Rose07, 4.3], a bit further can be addressed on that matter now based on industrial observations for why FPGAs still forge ahead with these sheer gaps. It can be seen that the gaps are very high and very different for the three categories. First the biggest one is in area, this somehow gets slightly compensated by the fact that FPGAs are always a few generations ahead in Moore's law and since they are programmable devices for large segment of clients so they have mass production that further helps to reduce the cost. For timing sometimes using the parallel nature of FPGAs some exploitation can be tried with tradeoff in Area and Power. However the critical aspect certainly lies in power consumption that is now considered a major challenge in ASICs and therefore surely becomes further prominent in FPGAs. This is the hardest challenge for FPGAs [FPGA06Panel, 2.9]. As the architecture of FPGAs is inherently made with huge flexibility of prototyping, it is relatively difficult for FPGAs to take full leverage of the advanced power management schemes used in latest SoCs like DVFS (Dynamic Voltage and Frequency Scaling), clock gating, GALS (Globally Asynchronous Locally Synchronous) etc. It is there where the *vs FPGA* can have an edge above FPGAs due to coarser grain nature and programming style that have some superior potentials to exploit the latest advancements in the above mentioned power management techniques. Because no matter how many Hard Macros FPGAs absorb in them they have to maintain their original dominating potential of prototyping and target a wide segment of market which makes them low cost.

Huge Compile Time

The high dominating potential of RTL programming makes the compile time of designs (Synthesis, Mapping, and Place & Route) on FPGAs much larger compared to software compilation of designs on processors. As FPGAs have immensely increased in logic density over time due to Moore's law this issue is now getting crucial for design space exploration of large designs on FPGAs. In general also for normal designs, designers often find it very time consuming for design space exploration of RTL designs on FPGAs due to their long compile time (inevitable due to fine grain hardware) compared to microprocessors.

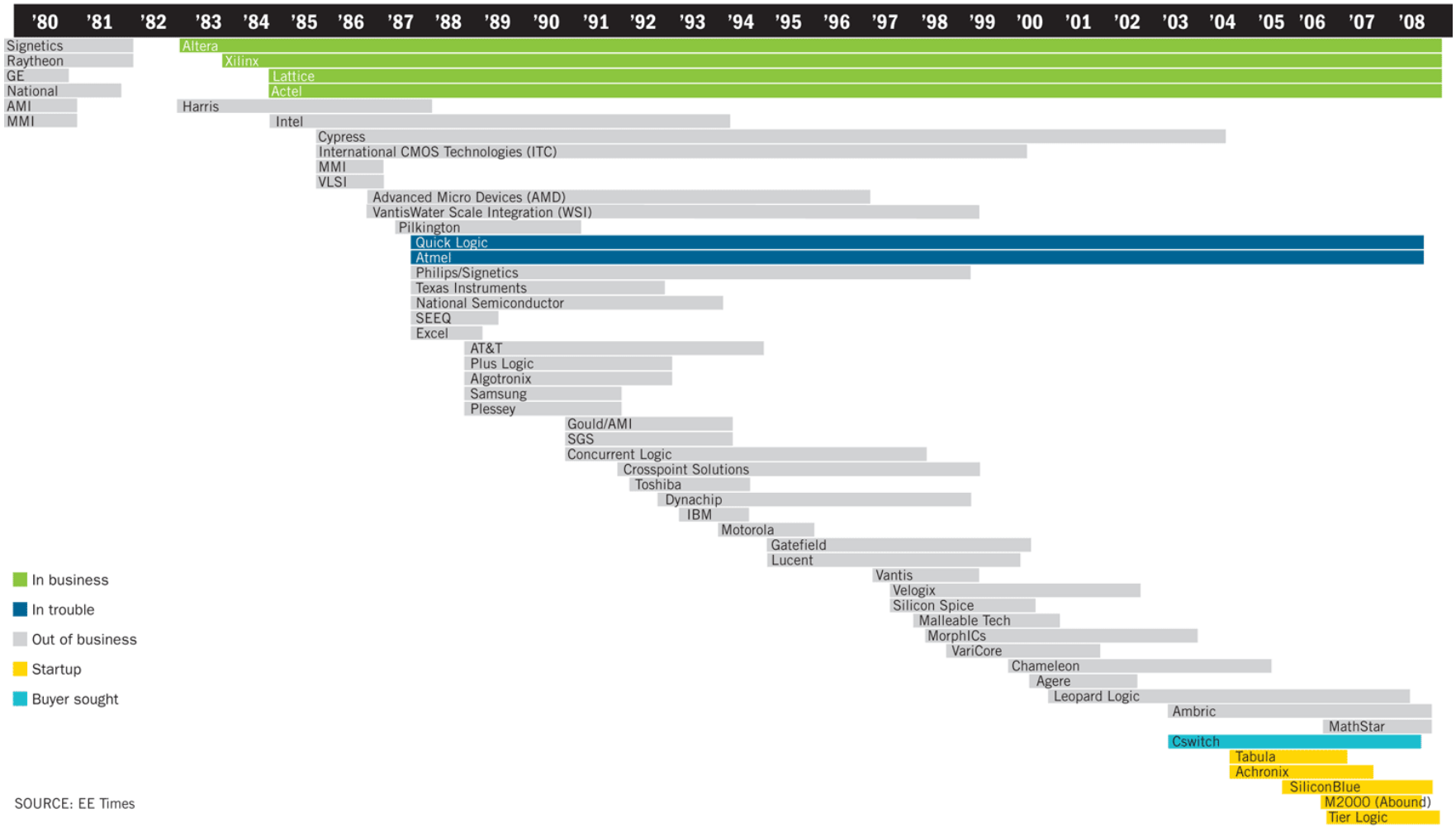


Fig. 2.41: History of Programmable Logic Device (PLD) startups

3- Survey of vs FPGAs companies

[Williston09, 2.13] presents an industrial survey of several MPPA, coarse grain companies, the list of the companies and their products are shown in figure 2.42. Some companies in the list are out of business and survey gives commercial details about the failure of them.

COMPANY	PRODUCT
Ambric	AM2000 family-344 Processors
Aspex Semiconductor (U.K.)	Linedancer 4,096 Processors
BrightScale	BA 1024 Video Processor
ClearSpeed Technologies	Multithread 96-element array processor
Coherent Logix, Inc.	hx2100 HyperX-based DSP
CPU Technology, Inc.	Acalis7 Field-Programmable MultiCores
Element CXI	Reconfigurable Array
Elixent/Panasonic (U.K.)	D-Fabrix Array
IMEC (Belgium)	ADRES: Coarse Grain Array for VLIW's
Intellasys	Scalable Embedded Array 24 processors
IP Flex (Japan)	DAP/DNA-955 16-bit processors
MathStar	Field Programmable Object Array
Motorola Labs	Reconfig. Streaming Vector Processor
NEC (Japan)	Dynamically Reconfigurable Logic Engine
PACT XPP Technologies	XPP 3C-64 processors
PicoChip Designs (U.K.)	picoArray Massively Parallel Array
Plurality (Israel)	Hypercore Processor: 16-256 cores
Rapport Inc.	Kilocore KC-256 with 256 processors
Recore (Netherlands)	Montium Tile Processors
Silicon Hive (Netherlands)	Moustique Block Accelerators
Stream Processors Inc.	Storm-1 Family-80 32-bit ALUs
Tabula	ABAX 3PLD
Tilera	TilePro36 & 64

Fig. 2.42: Industrial survey of MPPAs and Coarse Grain companies [2.13]

4- Why vs FPGAs mostly failed/fail

It is widely known that majority of the vs FPGA (also new FPGA startups) failed to succeed or found very limited acceptance. This section tries to briefly highlight the possible reasons based on knowledge gained from this thesis work and industrial expert's opinions. The above sections discussed the fundamental pros and cons of FPGAs. If one try to analyze from technology stand point (setting aside the more prominent business realities) the fundamental pros of FPGAs have usually been the fundamental cons of vs FPGAs (coarse grain in particular) solutions. As most of these solutions were novel and due to obvious programming complexity issues had to create some new or flavored versions of standard languages (often C) for their compilers to take efficient use of the hardware. Such solutions can be good for research projects or some very specialized applications but it is virtually impossible for them to find wide spread acceptance in industry due to lack of standards behind them. The companies see them as a risky investment because almost all the source code is wasted if want to switch to another provider. Furthermore industry is highly IP dominant, FPGAs enjoy benefits of IPs as they are mostly RTL based. For vs FPGAs solution it is very hard to take benefit of that. Furthermore from software IP point of view (as vs FPGAs are mostly software programmable), the software is always in standard languages and furthermore the legacy code has a very crucial benefit in industry and is mostly behind the dominating monopoly of a processor. Furthermore from business point of view [2.13][2.14] have highlighted that the quality and particularly cost of the offered EDA tools for programmable solutions in many cases is more prominent reason of failure than the hardware. So the EDA tools of programmable devices have to be ideally free or very low

cost compared to the cost of the silicon tools (this also explains why even leading giants like Xilinx, Altera provide a full tools suite free or at a fraction of cost of 3rd party tools for clients, making it not obligatory to use 3rd party tools, hence ensuring that tools cost does not create an entry barrier for lower end clients).

5- New FPGA startups, their differentiation

This section gives a short overview of some new FPGA startup companies. It focuses on their key differentiation to find niche among FPGA market space which is highly dominated by the Big-2 (Xilinx, Altera). As was discussed above that programming has often been the reason of failing of several solutions. It is worth noticing among all these new startups that they all use standard RTL flow even if some have un-conventional solution. Other noticeable thing is that TSMC is the FAB for almost all of them. That is making TSMC almost a de-facto fab of all the FPGAs. Actel, and Altera are well known for TSMC and Xilinx also has moved to TSMC HighK Metal Gate (HKMG) for its 28nm devices.

Abound Logic (www.aboundlogic.com)

Abound Logic (former M2000) is focused on creating low cost ultra high density FPGAs for prototyping, high performance computing and telecom applications. Their current proposed Raptor FPGAs [1.34] are built on TSMC 65nm and have logic densities in range of 750K LUTs, with numerous hard macro blocks of memories, DSPs, SerDes, Ethernet, PCI etc. The company is also known for providing the embedded FPGA FlexEOS macros a few years back when it was M2000. Their eFPGA is found in the Morpheus project [2.28]. According to their website it seems that they no longer support eFPGAs. Now their focus is fully on developing configurable logic SoCs. Some notable architectural patents are referred in [3].

SiliconBlue (www.siliconbluetech.com)

SiliconBlue has a major focus on low-power FPGAs which can be used for battery-based portable devices. Their iCE65 FPGA devices family is built on low-power TSMC 65nm process. They have done several innovations in packaging and configuration mechanism to make their devices very compact, low-power and single chip solution for the target. Their FPGAs have very low static power. Their FPGAs compared to other providers are relatively small, logic cells (LUT4+FF) range from 1200 to 16000. They have embedded memory blocks and phase-lock loops (PLL) as hard macros. They also propose their FPGAs in the form of a Die for SIP (System in Package) solutions. One of their most appreciated innovations is a single chip solution using embedded non-volatile XPM memory from Kilopass which loads the configuration to SRAMs of FPGA on power up [1.33][1.34].

Achronix (www.achronix.com)

Achronix is the first FPGA to be commercially launched which is different from conventional architectures. They have developed Asynchronous FPGAs. This allows very high speed operations. According to the company they claim to deliver world's fastest FPGAs with frequencies up to 1.5GHz. Their Speedster family of FPGAs is fabricated on TSMC 65nm process [1.36]. Their logic densities go up to 164K LUTs. In addition they have hard blocks of memory, multipliers, SerDes, PLLs and memory controllers. Their CAD tools suite ACE (Achronix CAD Environment) provides a seamless classical RTL tools flow for the programmer by hiding all the effects of Asynchronous FPGA hardware. Their target market segments are networking, telecommunication, DSP, high performance computing, military and aerospace etc. The company has got special attention of industry in 2010 when they announced partnership with Intel to make 22nm FPGAs on Intel process [1.37], that is exceptional as Intel is well known in industry to never/rarely share its process technology. Some notable architectural patents are referred in [3].

Tabula (www.tabula.com)

Tabula's technology can be considered as a masterpiece of dynamic reconfiguration. The device is not physically 3D in manufacturing; they call the time as 3rd axes. By this advantage their ABAX 3PLD devices fabricated on 40nm TSMC [1.39] process when compared to an equivalent classical 2D FPGA have gains of around 2.5X in logic density, 2.0X in Memory and 3.7X in DSP performance. More importantly as stated before, despite of the architecture which is completely un-natural physically, the programming model according to the company is purely standard RTL based. Their 3D Spacetime Compiler makes this possible. Some notable architectural patents are referred in [3].

Tier Logic with 3D FPGAs (Company has Folded)

Tier Logic (in July 2010 the company has closed business and www.tierlogic.com is no more available!) actually fabricated a 3D device. Their groundbreaking innovation was that they completely removed the configuration SRAM cells from the silicon and implemented them on top of the metal layers using Thin Film Transistor (TFT) technology from TOSHIBA. This allows almost doubling the logic density of the FPGA fabric as the millions of SRAM configuration cells are no longer there. This also helps in increased performance and power saving. The major benefit they proposed lied in the fact that the configuration is on the top of device, it is therefore very easy to transform the device to an ASIC (Structured ASIC concept) by just removing the SRAM layer and replacing it with programming the metal layer while keeping same timing [EETimes10, 2.45]. Some notable architectural patents are referred in [3].

6- New Trends among top FPGA vendors

FPGAs vendors have continuously kept evolving their devices based on market needs and competitor solutions potentials. Some notable new trends among top vendors can be characterized as follows.

Hard Processor: Although the concept is not new but FPGA vendors are particularly focusing on this direction, not just for putting a hard block of processor but to take the leverage of its legacy code dominance and rich IP-ecosystem. The newest solutions of Xilinx (28nm EPP Zynq-7000 series devices) and Actel (SmartFusion) devices were discussed in section 2.13. Having a Hard ARM processor inside the FPGA brings the processor dominance of ARM in embedded systems and its rich IP-ecosystem around AMBA [Xilinx10_Lysaght, 2.5]. Furthermore the hard processor (regarding huge FPGAs vs ASICs gap) is lot more silicon efficient compared to equivalent soft implementation of it. Altera is also following similar dimensions [Altera, 1.27] [EEtimes10, 2.49][EEtimes10, 2.50].

Interest in ESL: It was discussed above in the fundamental pros and cons of the FPGAs that compile time is getting significant in FPGAs due to the RTL flow. It has also been often felt that the RTL design flow of FPGAs present an added knowledge challenge for software designers who are more acquainted with C/C++. Furthermore in most cases design and verification of RTL is much longer and complicated compared to software. FPGA vendors are specifically focusing on the potentials of ESL to bridge this gap. Although it does not solve any compile time issue, but gives a step of ease for the designers to program their designs on FPGAs like software. The ESL tools allow a medium with full backend support to write the applications in more familiar and easier ANSI C/C++ and the tools automatically transform them into RTL that is then mapped on FPGAs with classical tools, making RTL code in principle like assembly language. With advancement of ESL tools rich in capabilities ESL is a prominent future direction among FPGA vendors [Xilinx10_Lysaght, 2.5] and they are significantly investing in this direction [2.51].

3D Stacking: To further improve Moore's law benefits beyond two-dimensional scaling, the die-stacking is getting common in industry. Xilinx 28nm devices use specialized form of die-stacking to build larger and heterogeneous devices with better yield control [Xilinx, 1.9][Xilinx, 1.10].

2.3.5- Industry is heading for Platform Collision

The above sections covered different types of programmable hardware. Many of them are competitive and getting more and more competitive to each other. This section will try to analyze this issue from future collisions perspective in the light of pros and cons of those solutions and what role eFPGAs can/will play in that scenario.

1- Cloudy Future

Previous sections covered several solutions that exist in industry (along with their market share) that can be treated as programmable hardware. Setting aside the high-end PC market that is a very specialized and huge portion of industry one can categorize three main dimensions/waves of solutions that are heading for collision for competitiveness, they are shown in figure 2.43. This section tries to analyze from technology stand point, they are miles apart in current market share (as we seen above) and that influence in many cases overrides technology in industry. It will observe, what are their dominating benefits vs each other.

FPGAs continue to go more and more heterogeneous to increase the competitive value compared to ASICs/ASSPs in several cases, they are transforming into programmable SoCs. The new trends of adding hard processors will further strengthen the move towards that direction. They will continue to increase in dominance by having range of customized products targeted for specialized domains. However it is relatively difficult for FPGAs to create customized solutions for every possibility. Their strength is largely in the horizontal business model which makes the devices low cost due to mass production of a narrow range of products.

The **ASSPs** (including MCUs) are becoming programmable platforms, they represent in principle opposite side of FPGAs spectrum. FPGAs are continuously getting heterogeneous to reduce the expensive fine grain tradeoff for differentiation functions only making them more and more competitive to ASICs/ASSPs with the power of inherent flexibility of FPGAs. ASSPs/ASICs are already built up with exactly the things that are needed, what they lack is a limited flexibility for differentiation. Processor with flexibility of software at present is central source of flexibility. In addition the trends for using programmable or configurable IPs are also prominent among them in various forms for some limited flexibility, for instance small portions of high density non-volatile memories IPs, specialized processors IPs, coarse grain like solutions IPs. *eFPGAs* clearly seem to be an interesting ingredient for future to compete with the inverse scenario of FPGAs.

Third is the emerging wave of primarily **vs FPGA** solutions. Currently their biggest challenge to create a significant competitive threat is the parallel programming crisis. This wave is further complicated by several similar competitive solutions within itself. A broad distinction can be made as MPPAs and coarse grain. They have some similar challenges; most prominent of them is programming crisis. However in this regard they slightly differ. Coarse grain historically suffered additionally because their problem to go forward was in most cases just their own problem of having good compilers and some language extensions to make them efficiently usable. On the other hand MPPAs are facing the challenge of parallel programming which is an industry wide challenge now as industry has moved to multicore era. This phenomenon can give an added benefit to them over coarse grain whose parallel programming crisis is similar but not same (coarse grain generally do not use processors for computing). Will coarse grain also get an indirect benefit of the efforts of parallel programming; it is difficult to say with certainty. But as the biggest force and undisputed king who is driving the industry at present is power consumption. In that regard potentials of efficient computing style of coarse grain cannot be neglected. MPPAs are very stylish and attractive to reconfigurable research community but the central processing component is the same old friend von Neumann machine which is famously known to be inefficient computing device [Hartenstein, 4.64-to-4.72]. *eFPGAs* are also highly attractive for MPPA like machines to bring the usual marriage of HW/SW co-design.

What seems highly visible is that industry is heading for platform collision with several players with different kind of solutions heading for same or similar markets (GPGPUs are also getting traction for several non-conventional solutions). Industry experts are already seeing this collision course ahead [EEtimes10, 2.17]. Who will win, loose, soon fighting survival battle is clearly difficult to understand and foresee. What is certain is that "*One Ring will rule them all, and that is power consumption*" [S-3b].

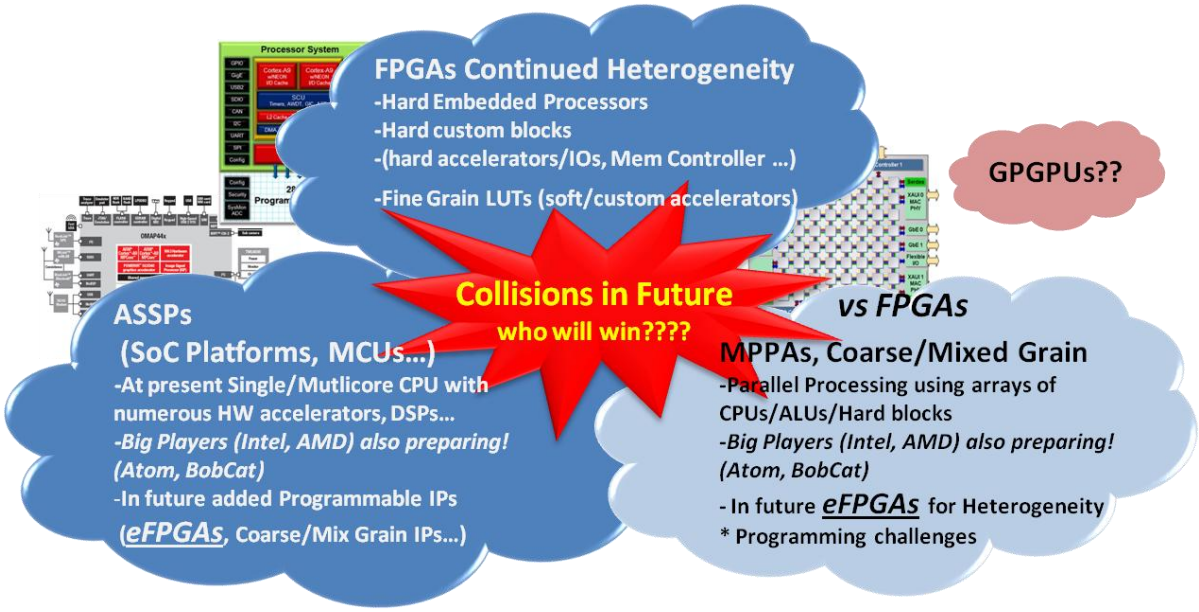


Fig. 2.43: Industry is heading towards Platform Collision [S-3b]

2- Is eFPGA a missing IP

The above discussions observed that eFPGAs clearly present great opportunities in several markets. It seems clearly a missing ingredient in the industry [S-1]. However then usual question mentioned before again rises, eFPGAs never succeeded in industry is well known. This clearly raised the motivation that "*should we learn from past or run from past*". Sometimes things change with time and sometimes things fail just because time was not right. Around 10 years ago FPGAs were not as prominent as they are now, the challenges of power and market pressures were not like they are now. Around 10 years ago Altera made Excalibur devices with Hard ARM processor, which was an industrial flop, and Altera abandoned Hard Processors for a decade (now planning again for 28nm devices [Altera, 1.27]). Similar thing was done/announced in 2010 (although Xilinx had Power PC for years, but the nature of new move is different) when two leading FPGA vendors announced devices with Hard ARM processor and entire industry is wowing at it to be the great innovation and need of the time. Intel Stellarton [Intel, 2.48] [EEtimes10, 2.49] (Atom + Altera FPGA system in package) is another prominent example of growing importance of FPGAs/embedded FPGAs in industry in context of scenarios discussed in figure 2.43.

There is a need to find ways, how to remove the barriers that kept eFPGAs difficult to succeed in past, this thesis work has tried to address some.

2.4 Summary

This chapter covered in detail the contribution R of this thesis work including the research motivation explorations for B and G (chapter 1), with discussions divided in three distinct sections that can thematically be divided into two parts. The first part (section 2.1 and 2.2) investigated in detail the research done in industry and academics regarding FPGAs and eFPGAs and second part (section 2.3) addressed in detail the major findings of industrial analysis to get a global picture of all the technologies, their status and potentials and trends in industry. The main findings and motivations they provide for research directions for this thesis work are described below.

General Findings

Section 2.1 investigated in detail the architecture fundamentals of FPGAs, observed the advances in state of the art solutions and academic research to find research challenges of FPGA architecture. It found importance of FPGA CAD for FPGA architectural research. It also investigated some beyond classics research efforts on/under way to address some architectural challenges of FPGAs.

Section 2.2 analyzed the present and past solutions in industry in the direction of eFPGAs. It investigated research efforts by academics in classical FPGA-like eFPGAs and some different forms of programmable solutions (coarse grain etc.).

Section 2.3 detailed the general industrial survey findings to see the global picture of the issues, find out pros and cons of different programmable technologies and what their status in industry is. It investigated the fundamental advantages of FPGAs compared to competing solutions which give them edge over them to find potentials of classical FPGA–like eFPGAs. It observed the changing trends in industry and the collision course it is heading towards to see if/can eFPGAs fit in that scenario.

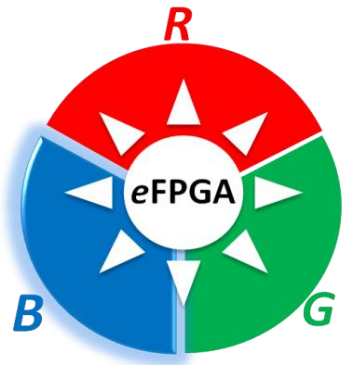
Conclusions for the thesis

Following are the prominent points learned from this chapter for the thesis regarding research work motivations and challenges for eFPGAs.

- Standards play a very important role in industry. The research work must focus on the importance of that issue e.g. regarding tools, programming, architectural implementation and integration etc.
- Power consumption has become a critical and challenging issue throughout industry
- eFPGA with a classical FPGA-like architecture is advantageous. It helps to leverage the dominant points of FPGAs (RTL flow, IP eco-system benefits, high flexibility etc.)
- eFPGA should be highly customizable and flexible enough to target several technologies instantly/easily (avoid having strictly fab/node dependent full custom solution)
- eFPGA must be efficient in architecture, small and highly customized (silicon gap challenge) for target demands to prove a good value proposition for SoCs
- User friendly infrastructure should be created to facilitate creation of customized eFPGAs with full software support to program them (adaptable CAD)
- Investigate potentials and challenges of eFPGAs in systems to understand their value proposition
- Investigate new/beyond classics technologies/methods that can enhance the potentials of eFPGAs

The next chapters address these general motivations. Chapter 3 presents the tools infrastructure which facilitates to create and explore customized eFPGAs. Chapter 4 will address in detail the architectural explorations and innovations for eFPGAs based on the knowledge gained from research survey of this chapter. Chapter 5 will discuss the efforts for eFPGAs in systems. It will also describe some beyond classics technologies perspectives to enhance the classical properties of eFPGAs.

Chapter 3: CAD Infrastructure of eFPGA



The research of FPGA architecture is virtually impossible without the CAD tools infrastructure for creating, exploring, programming and validating different architectures to find better architectures as was discussed in chapter 2. Since this work is targeted for embedded FPGAs (eFPGA) it is essential to have the entire FPGA CAD infrastructure (as eFPGA has FPGA-like classical architecture) to program them. Furthermore due to being eFPGA brings additional challenge and opportunities to create customized eFPGAs targeted to specific needs. This objective in general is similar to all FPGA vendors but is particularly more crucial for eFPGAs as the target market segment can/may be narrower compared to FPGAs which even if are domain specific are targeted for a relatively wider range of applications in that domain to keep their benefits of horizontal model which makes them low cost with mass production (chapter 2). With eFPGAs it is possible to further fine tune the architecture based on end market/client needs to have better silicon tradeoff due to adding eFPGAs. So other than the programming tools it is important to have powerful user-friendly infrastructure to facilitate creation of efficient eFPGAs and programming tools to adapt to any architectural changes and customizations done.

This chapter primarily is an overview chapter for the thesis work describing the CAD tools infrastructure of eFPGAs. This thesis work is more focused on hardware, and programming tools are not the main contribution of this thesis work as was described in chapter 2, this work contributed in the motivation of the adaptable nature of programming tools and use them for architectural exploration and indicating rooms of new innovations for programming tools to address architectural enhancements (the usual FPGA co hardware-software challenges). Section 3.1 provides the basics of eFPGA architecture explored in this thesis work, knowing them are essential for discussions of this and next chapters. Section 3.2 explains the adaptable eFPGA programming tools suite of Menta (eFPGA Programmer), which provides standard RTL flow for programming the eFPGAs. Finally section 3.3 describes in detail the infrastructure of graphical tools suite (eFPGA Creator), which facilitates to create and explore eFPGAs. The key contributions of thesis work are outlined below, deeper details about the tools, how they are made and work are proprietary of Menta and are beyond the scope of discussions.

Contributions of thesis work: eFPGA Creator is a large infrastructure with several set of tools and components. The whole tools suite is not the direct contribution of this thesis work; however a brief overview of the whole tools suite is presented to facilitate the ease of discussions. The thesis work is heterogeneously involved in several aspects/design of the tools/sub portions which cannot be easily separated and explained. The prominent and coarser gained contributions of this thesis which are either direct contribution or a base contribution (upgraded by Menta) can be categorized as.

- General motivations for tools, flows, potentials of eFPGAs and how to exploit them [S-1][S-7]
- Architectural exploration infrastructure in Library and Architecture managers, component based design philosophy and its benefits/challenges/potentials
- Hardware generator (automatic VHDL and basic scripts for implementation on target technology)
- Foundation of analyzer system and its tools to facilitate and accelerate research/exploration [S-9]
- Industrial overview and potentials/challenges (technical and commercial) of eFPGA Creator for industry [S-6]

3.1 Architecture Fundamentals of eFPGA

This section provides basic fundamentals of eFPGA architecture that is investigated in this work. Although this chapter is primarily focused on CAD infrastructure (next chapter is dedicated for architectural research) explaining the basic fundamentals here facilitates the discussions of this and the next chapter where they will be addressed in detail.

3.1.1 Island style uni-directional routing Architecture

Figure 3.1 shows fundamental diagram of eFPGA (homogeneous in this case) architecture. It can be clearly seen it is classical *island-style* architecture. As was stated in chapter 2 the commercial names of components vary among all the companies but fundamentals remain almost the same. For the case of eFPGA it can be observed that eLB (will be discuss in next section) represents the logic block or CLB of scientific literature. An important point to note is that eLB is directly connected to the SB, compared to the *connection box* approach of which is common in other literatures, as was observed in chapter 2 (shown in more detail in figure 3.3 for illustration); the switch block performs all the switching (in principle it is similar as was discussed in chapter 2). The eFPGA has *unidirectional* routing architecture achieved through multiplexers. Furthermore eFPGA is pure soft (written in VHDL) and technology independent. There are no pass-transistors, tri-state buffers, SRAM cells etc. The routing and configuration architecture is built up with standard multiplexers and Latch/FF of standard cells library of the target node to which the RTL of eFPGA is ported hence giving technology independence of soft eFPGA.

Key Motivations to select this style

Motivations like island-style, uni-directional routing etc. are well known for FPGA researchers as was discussed in chapter 2. Following are some of the additional key motivations to investigate eFPGA architecture in this direction (figure 3.2). Chapter 4 will address these points in more detail.

- *Soft eFPGA target*: Since the motivation is to create soft eFPGA (technology independent, easy integration in SoC with silicon tool flows) with multiplexers based switching, this type of architectural configuration provided a relative ease of design compared to classical architectures (figure 3.3) which are primarily based for transistor based full custom designs
- *Uni-directional routing*: The uni-directional routing architecture as was observed in chapter 2 (figure 2.1) already merges the output CB with SB for creating single driver architecture
- *No Local Interconnect*: The eLB is directly connected to SB, The input for every BLE (classical LUT+FF element) comes from SB and their outputs go to SB (feed-backs are done through SB as shown in figure 3.2). The conventional local interconnect (LI) depicted in figure 3.3 inside the logic block while provides the superior benefits for routing by providing the logic equivalence for the inputs and outputs of logic block (LB) as was discussed in chapter 2 (figure 2.26), is highly penalizing in terms of area, its depopulation or complete elimination is a strong research area of FPGA architecture research (see patents appendix A1 for details of state of the art in this area)
- *Relatively new what-if research contribution*: The classical island-style architecture for figure 3.3 is well addressed and explored. This partially new type of architecture is also good for research contribution by providing some new what-if investigations results
- *Unified SB (no CB)*: The unified SB with CB (input and output) merged in it also provides a research ease by focusing entirely on SB as it performs all the switching and joint optimization potentials can be investigated (chapter 4 will address it in detail)
- *Easier Tile/Component handling*: Having an architecture of the form of figure 3.2 reduces the amount of components to build a tile and managing them (this point will get explained in eFPGA Creator discussions in later sections)

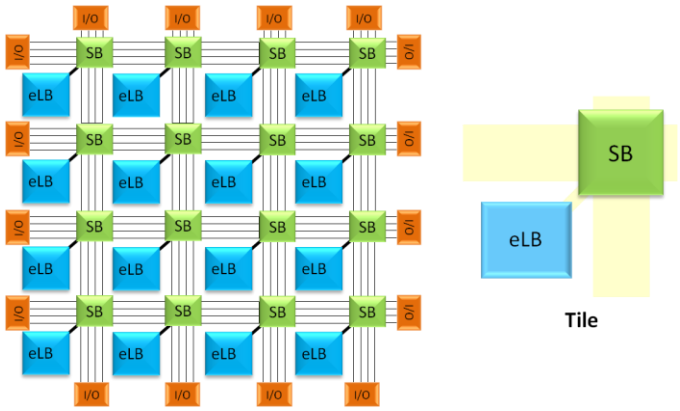


Fig. 3.1: Fundamentals of eFPGA Architecture

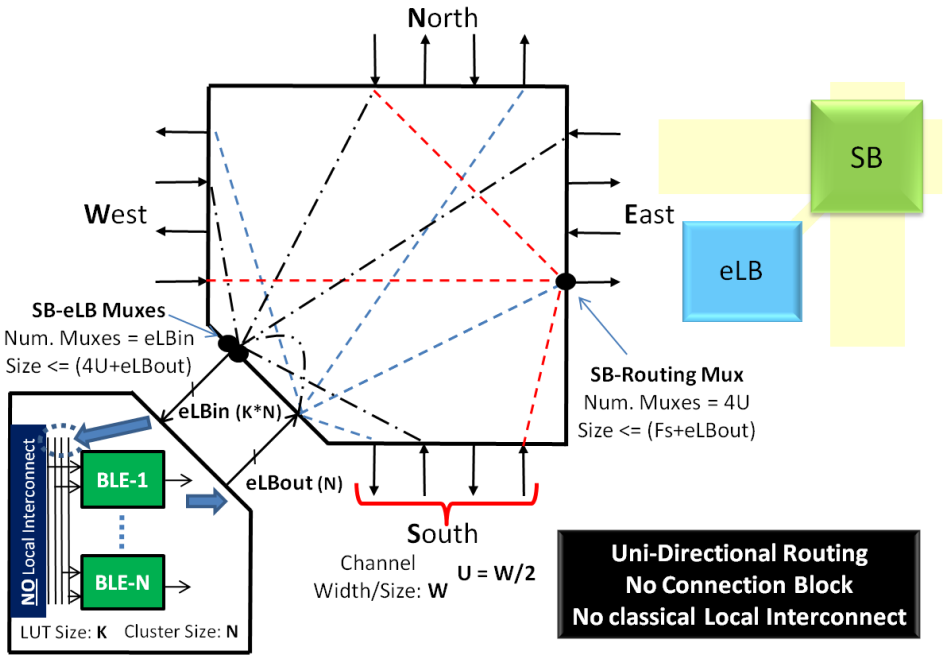


Fig. 3.2: Details of eFPGA architecture fundamentals

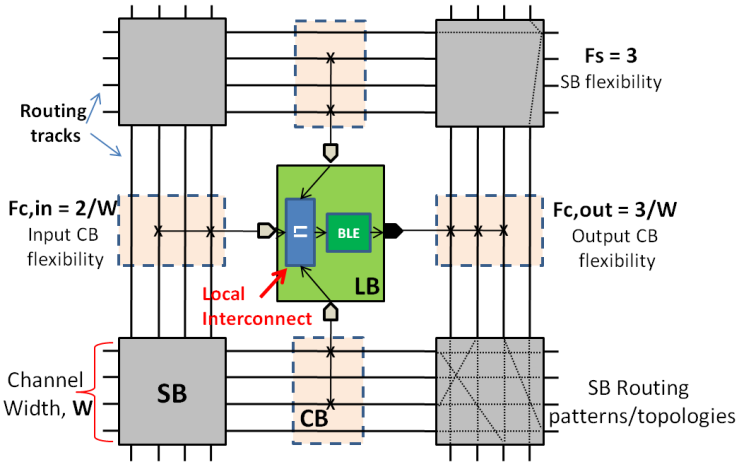


Fig. 3.3: Fundamentals of a classical Mesh architecture ([Betz&Rose99, 4.1]) for comparison

General Architecture Parameters

Following are some fundamental architecture parameters of eFPGA in relation with figure 3.2. They will be explored in detail in chapter 4.

***LUT size*:** The size of LUT inside the BLEs. It is often represented by K in literature; this thesis will use the same for consistency.

***Cluster size*:** Number of BLEs inside eLB (logic block). Often represented as N, same will be used here

***Channel width*:** The size/width of routing tracks; It is often represented as W, same will be used here, in addition since the architecture is unidirectional. For this work another variable U is used for experiments in this thesis, which is W/2 (the thesis use equal number of input and output tracks in exploration)

***SB parameters*:** The SB (in connection with CB) has multiple architecture parameters in scientific literature to characterize the property of architecture [Betz&Rose99, 4.1]. Some key elements are highlighted in figure 3.2 and 3.3 for instance: Fc,in (number of routing tracks LB input pin connects to), Fc,out (number of routing tracks LB output pin connects to), Fs (flexibility of routing tracks in SB for switching) etc. Chapter 4 will address them in more detail in advanced discussions and explorations.

3.1.2 Building Blocks of eFPGA

Figure 3.4 illustrates conceptual overview of the five building blocks of eFPGA Core architecture interconnected by the routing architecture trough the unified fundamental block SB (not shown for simplicity) which was discussed in previous section, and connected to the outside world by I/Os units. The well known benefits of island-style architecture are visible in it. Since eFPGA is an IP this gives an added opportunity compared to device FPGAs to decide the quantity, parameters and position of these blocks based on target needs. While explorations with only homogeneous eFPGAs are discussed in this work (chapter 4), the tools and work done by the thesis is already done to adapt to these heterogeneity of building blocks to build heterogeneous customized eFPGAs in future research. A brief description of the five building blocks is described below to give an overview of their motivation and purpose.

Fig. 3.4: Building blocks of eFPGA Architecture

eLB (embedded Logic Block): The eLB is the foundation building block of eFPGA Core that provides the foundation fine grain flexibility. It is the classical logic block component of an FPGA. The parameters of elB (LUT size, cluster size etc.) are customizable. Chapter 4 will address it in detail.

eMB (embedded Memory Block): The eMB is the embedded SRAM memory blocks to provide the hard memory blocks. The quantity (number of eMB blocks), position (in middle, corner etc.) and properties

(single/dual port, data width, size) can be customized based on target needs. eMB are target node dependent due to obvious reasons. The Fabs (including 3rd party companies) provide several options for these blocks and their standard cell libraries to be easily integrated with standard RTL flow.

eAB (embedded Arithmetic Block): The eAB is the basic arithmetic block. It can be a simple ALU, multiplier, DSP etc. Like eMB their properties, quantity and position can be customized based on target needs.

ePB (embedded Processor Block): The ePB provides the benefit of having a small microprocessor in the eFPGA Core. Although a soft implementation of processor can be done using eFPGA programming resources, the value proposition is much higher for putting a hard processor block (particularly from a 3rd party, that is often the case) to have the silicon tradeoff benefits as putting a soft processor on a soft eFPGA will/can be expensive.

eCB (embedded Custom Block): The eCB is most distinguishing block of eFPGA Core. Since eFPGA is an IP it brings the opportunity compared to device FPGAs to put some dedicated customized blocks based on target needs to have benefits of bridging the usual heavy silicon gap of programmable solutions compared to hardwired implementation. eCB can be components like complete cryptographic cores, PLLs, SerDes, Memory controllers, I/O controllers etc. Like ePB, eCB can be from 3rd party or SoC vendor integrating eFPGA. The infrastructure of eFPGA tools facilitates to integrate these blocks in eFPGA architecture.

3.1.3 eFPGA Core

As was discussed above, the architecture of eFPGA is highly customizable and target independent due to soft nature of eFPGA (written in VHDL), based on that there are two main possibilities of eFPGA Core. The pure soft version (this thesis work is focused on that) which can be ported to any process or the custom implementation (layout) of the logical design of soft version. They are briefly described below.

eFPGA Core-S (Synthesizable): The pure synthesizable version gives the highest flexibility for implementation due to target independence. Written in VHDL it allows immediate portability to any process node or fab. This feature alone has its own technical and commercial benefit. Chapter 2.2 discussed some past approaches where the provided eFPGA was on a specific process node, creating like that leads to higher silicon efficiency compared to soft approach. But it restricts the user to a specific node or fab which may or may not be easy or feasible in many cases for the SoC vendor to switch his entire design to another process just to add a programmable IP. Having a soft eFPGA removes these barriers. Furthermore since eFPGA is highly customizable and of small sizes, in many cases the silicon tradeoff with hard implementation can be balanced with the benefits which come due to soft implementation.

eFPGA Core-H (Hard): The hard implementation of eFPGA transforms the soft version to a specific node with custom design (layout) to highly increase the silicon efficiency. This option is ideal for mass production where the ROI (Return On Investment) justifies creating an optimal silicon implementation. Hence the concept of soft and/or hard eFPGA helps to bridge some of the past barriers for eFPGAs to be used in SoCs to provide the obvious benefits (Chapter 5 will further address them).

3.2 eFPGA Programmer™: eFPGA Programming tools suite

Figure 3.5 shows the flow of eFPGA Programmer tools suite which provides the standard RTL flow for programming the eFPGA. eFPGA Programmer is built keeping both technical and commercial considerations in perspective. It was observed in chapter 2 that programming tools of the programmable solutions have a deep impact on the success of a solution and the programmable solution providers must have low cost (ideally free) and good quality tools for their products. eFPGA Programmer is built on same philosophy, with taking further benefit of using 3rd party Synthesis tool instead of creating one due to

being eFPGA! (it will be integrated in a SoC so high quality 3rd party Silicon tools will already be available) hence focusing more on back-end.

eFPGA Programmer itself is not a contribution of this thesis, it was built by CAD team of Menta in close contact with thesis work which extensively use this tool for applications mappings and architectural explorations (chapter 4). An overview of the tool is presented to facilitate discussions. Below brief explanation of front-end and back-end of eFPGA Programmer is presented. Deeper details are beyond scope of this work.

3.2.1 Front-end

The front-end of the tool uses Synopsys Design Compiler (DC) for synthesis. It is not the part of eFPGA Programmer from licensing point of view. However for the reasons discussed above this does not cause any problem (license cost burden on clients issue discussed in chapter 2) as eFPGA is not a device but is integrated inside a SoC and the high-end de-facto silicon tools will already be available and can be easily used at the front end of eFPGA programmer, providing the strong VHDL/Verilog or mixed design entry ease. The synthesizer maps the design on the generic technology library of Menta (MGTECH) and the RTL netlist is forwarded to backend (eFPGA Programmer) which performs all the conventional FPGA CAD flow for mapping the design on eFPGA.

3.2.2 Back-end

The back-end of eFPGA programmers has familiar FPGA design flow. The mapping portion of the tools suite is based on enhanced ABC tools from Berkeley University [4.83][Brayton&Mishchenko, 4.86] which performs further logic optimization and LUT mapping of the synthesized netlist and pass on to proprietary clustering, placement & routing (PAR) tools. The architecture files provide all the information to perform these steps. The tools are highly adaptable to architecture changes and customization through the architecture files which are generated by eFPGA Creator (details in next section) hence providing instant programming tools (adaptable CAD) support for created eFPGA.

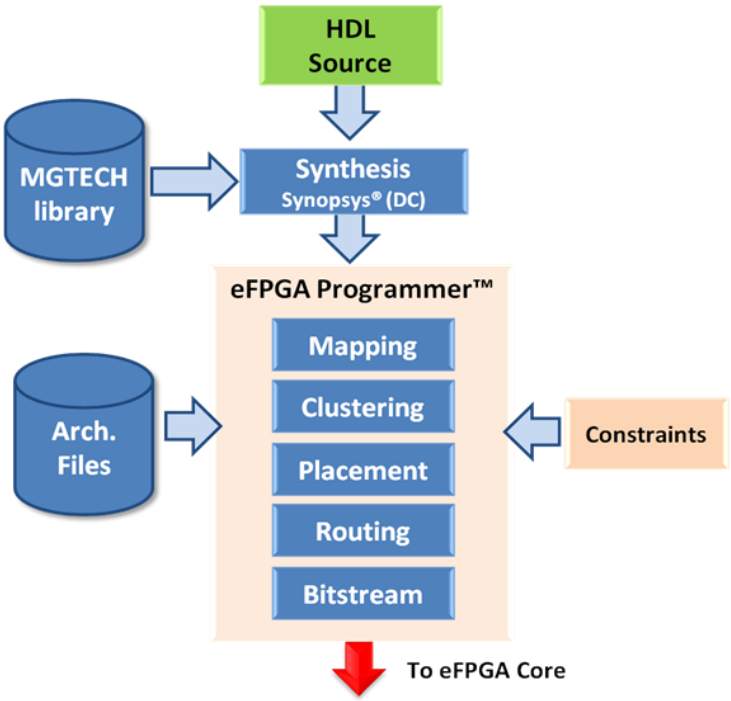


Fig. 3.5: eFPGA Programmer Flow

3.3 eFPGA Creator™: eFPGA Creation GUI tools suite

This section presents an overview of the eFPGA Creator tools suite which facilitates creation and exploration of customized eFPGAs. It first describes the general motivations and global overview of the tools suite. Then later sections explain the major components of tools suite.

3.3.1 Motivations & General Overview

This section provides general motivations and overview of eFPGA Creator tools suite. It first describes the motivations behind creating these tools, what were some of the inspiration sources and a brief overview of the tool which will be discussed in more details in the rest of this chapter.

1- Two objectives: Creation of eFPGA, Exploration of eFPGA

The discussions in this thesis have frequently used the term customized eFPGAs, efficient eFPGAs in the text until now. What is a customized eFPGA? The general guess that comes in mind is architectural customization i.e. changing LUT size, cluster size, channel width, I/Os, dimensions of array etc. based on target needs. Indeed it is true and a major goal of eFPGA Creator is that same foundation areas but they are not the only one!. The philosophy of eFPGA Creator is closely tied with the philosophy of soft eFPGA. This makes eFPGA Creator in some aspects similar and in some aspects different than other works e.g. VPR, which is more focused on exploring hard FPGAs with custom layouts that is common in all commercial device FPGAs, but at same time significant part of VPR's architectural exploration applies to all FPGA/eFPGA *implementation independent* architectural explorations.

There are two fundamental objectives of eFPGA Creator that are both tightly interlinked yet distinguished from each other. This point is important to note as it is slightly complex. The first objective is to create efficient customized eFPGAs based on the available set of library of the available components of eFPGA Core which are pre-designed, pre-verified and in some cases even silicon validated by test chips. [S-6] addressed this philosophy in commercial perspective; it will be discussed in more detail in next sections. The second objective is the architectural exploration part which helps to create these comprehensive libraries which are collection of several best cases and types which can either directly be suitable for target eFPGA or a good start point to narrow down the design space exploration direction. Having these things in mind it can be easily understand how closely these two things will be linked with each other yet different from each other particularly from a commercial stand point. These things will get further elaborated in later sections when the flow and main blocks of eFPGA Creator will be explained.

Based on these objectives the next biggest questions are how to make this entire infrastructure, what similar examples exist in industry or academics and are there graphical tools which facilitate creation of user friendly tools. These points are briefly discussed below.

2- Inspiration from VPR: ease of "what-if" experiments

The VPR tools [Betz&Rose99, 4.1] are the most inspiring and de-facto in FPGA architecture research community. The major benefit and motivation behind these tools is to facilitate the task of an FPGA architect to explore wide range of "what-if" architectural exploration by specifying the architecture parameters at a higher level [Betz&Rose00, 4.11]. With the description of some key parameters in architecture file, VPR creates the architecture with automatically resolving problems in the generated architecture if some conflict or bad choice arise due to unspecified architecture parameters [Betz&Rose00, 4.11] (as only key parameters are specified in architecture file), hence creating an architecture ready to be experimented by implementing benchmark applications. The VPR technology was acquired by Altera in 2000 [1.24] and is foundation of their proprietary FPGA Modeling Toolkit (FMT) which enhanced VPR to deal with state of art FPGAs complexity with first successful use to design Stratix™ architecture [Altera_Lewis05, 1.22].

The "what-if" experiments ease and the corresponding adaptable CAD provided nice inspiration for eFPGA Programmer and eFPGA Creator, doing similar thing but in a bit different way with partially similar and partially different objectives. It will be discussed in more detail in later sections.

3- Inspiration for GUI based tools: added user friendliness dimension

Graphical User Interface (GUI) other than the obvious aesthetics provides great help and boost in productivity of several problems. While conducting a research work it is often desired by researchers to have some graphical tools to facilitate and accelerate their research work. However it is like industrial ROI (Return On Investment) form of scenario. There is a tradeoff related with such approaches to judge the amount of time spent on creating them compared to benefits in return. Over the years the tools to help create GUIs have greatly evolved and have reached a state that it is becoming easier for researchers to quickly create fine GUIs with complex features to aide in their experiments. As this thesis work is industrial so the inherent commercial scenario related with GUI based tools provided further need and motivation to create GUI based infrastructure. When one considers about GUIs, two prominent names come in mind that are in wide spread use and need no introduction, Eclipse (www.eclipse.org) and Qt (qt.nokia.com). Qt was selected as it better served objectives and needs, furthermore its C++ programming model further helped to work and integrate with it. Next sections will present the tools suite and it will be seen how facilitated the flows and tools are due to GUI, serving the two objectives of eFPGA Creator (Creation and Exploration of eFPGAs).

4- Global Overview of the tools suite

Figure 3.6 presents theme diagram of eFPGA Creator, providing the global overview of the tools suite with its two objectives/flows discussed above. To better explain the flows and how they differ, it is essential to get basic idea of building elements of eFPGA Creator which are briefly described below and are explained in detail in next sections.

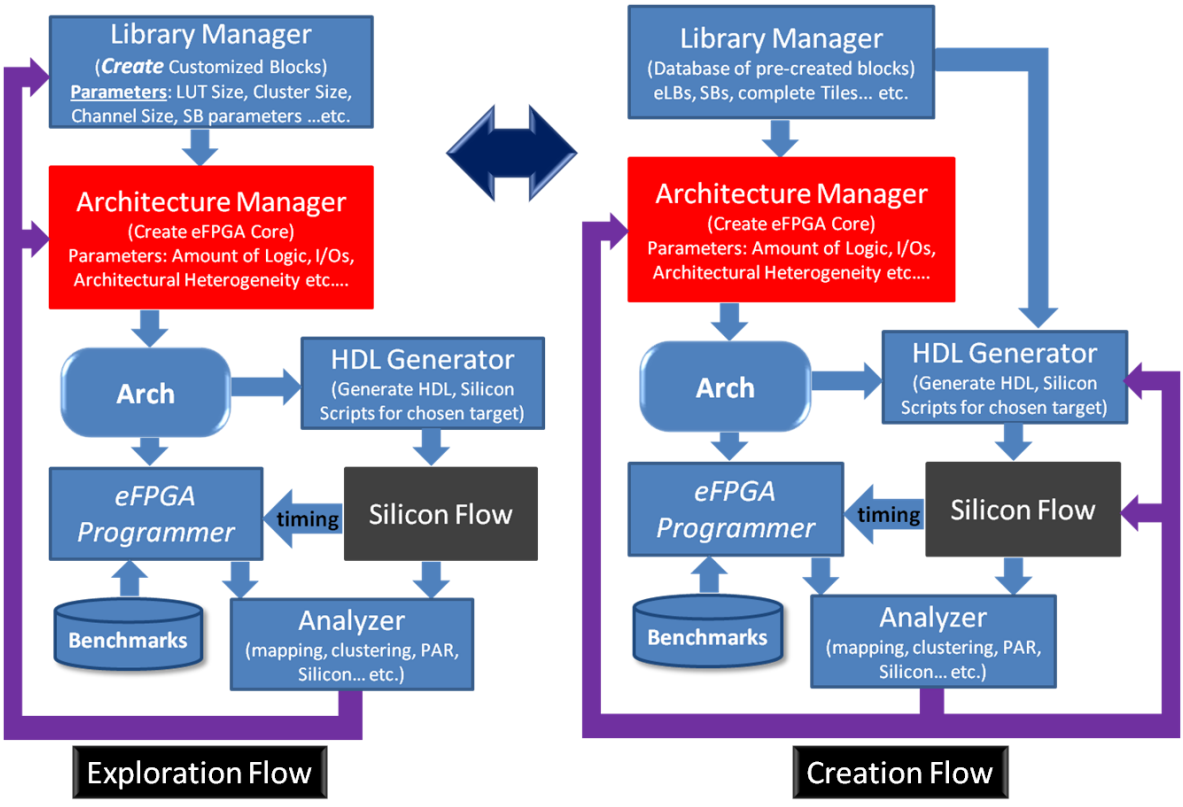


Fig. 3.6: eFPGA Creator Tools suite global overview

Library Manager

The library manager is like a central database of all the components that can be used for creating or exploring eFPGA. It is composed of components like eLBs, SBs etc. with different architectural parameters (like LUT size, cluster size, channel sizes, SB topologies etc.). It can be considered like a kind of standard cell library of eFPGA components with which the eFPGA Core is built. Section 3.3.2 will discuss it in detail.

Architecture Manager

Architecture manager builds the architecture of eFPGA Core using components from the library of library manager. It can be considered as a database of architectures. It creates the hardware of the core for silicon implementation and corresponding architecture description files for eFPGA CAD tools (eFPGA Programmer) for implementing applications. Section 3.3.3 will discuss it in detail.

HDL generator (Silicon Manager)

The HDL generator creates VHDL files, front-end and back-end scripts for implementing the eFPGA RTL on a selected process. HDL Generator is presented as a separate entity in theme figure for ease of discussion and illustration. All these tools and managers are highly interlinked in terms of design and working which will be illustrated in later sections.

eFPGA Programmer

eFPGA Programmer is the programming tools suite of eFPGA and was discussed it in detail in previous section. It maps the benchmark applications on the target eFPGA. The corresponding architecture files (including silicon information) provide all the architectural and timing information needed to map applications on the created architecture.

Analyzer

The analyzer tools help to facilitate analysis of benchmark applications on the target architecture. This provides a decision stand point if the architecture specifications are meeting the requirements and if not what will be the better tuning direction to optimize the architecture (architecture parameters, architecture heterogeneity, silicon implementation etc.). Section 3.3.4 will discuss it in detail.

eFPGA Creation & eFPGA Exploration

With the basic understanding of the basic blocks in figure 3.6 described above, the theme of two objectives (can be considered as two modes/phases of eFPGA Creator) can now be explained with the help of figure 3.6 and can be seen how closely these two aspects are interlinked yet somehow distinct.

During Exploration more emphasis is put on exploring new concepts and better architecture components (this part is much similar like FPGA architecture research) in this case relatively moderate stress is put on silicon implementation as main objective is to analyze new architecture concepts and get an approximate idea about their silicon properties (area, power, speed etc.) to judge the benefits that can be achieved with those architecture innovations. The objective is to create efficient components database for the library.

During Creation the eFPGA creator creates an eFPGA architecture using the available components in the library manager which are pre-analyzed and in ideal case can even be silicon proven (test chip etc.). In this phase the emphasis is not on exploring architectural innovations of components (like changing properties of logic blocks, routing architectures etc.). The core is created/re-created using the available components in the library to match the target requirements (the process is facilitated by the analyzer), more stress is put on

silicon implementation in addition to the built-in help of eFPGA Creator, to build more optimal solution in a product scenario.

It can now easily be seen how interlinked these two phases are and in reality a mix of both is/can be used which is illustrated in the figure by interlink of the managers in two phases. Next sections will further illustrate that. At present eFPGA Creator has relatively stronger focus on Creation compared to Exploration. However Exploration part although being in continuous evolution and development is already built up to a level that it continues fueling innovation for creating new ideas and improvements to Create efficient eFPGAs. Chapter 4.2 will address the detailed experimented conducted using these tools.

3.3.2 Library Manager (eFPGA components creation)

The library manager provides an infrastructure to create a central database of building components/blocks of eFPGA architecture. It can be considered like a kind of standard cell library of eFPGA components with which different eFPGA Cores can be built. The tool consists of several sub tools which facilitate creation of customized components and their silicon implementation. A brief description of the flow is explained below and then the major components of the flow are discussed in more detail with snapshots of some of the portions to provide a feeling about the tools.

Figure 3.7 describes the global overview of the flow of library manager. It starts with the GUI editors which can create a new or edit an existing component in the library. The customization of components in particular SB-eLB (Tile) is done with user friendly GUIs giving the power to create different kind of "what-if" customizations to experiment the behavior. The created or edited component is stored in the library which makes it available for architecture creation or exploration. In addition to that the library manager has also built-in tools to generate VHDL of the designed component (Tile) along with all the silicon scripts for a selected process (In experiments presented in this work 65nm ST process is used) making it very easy for the researcher or hardware designer to quickly implement the component on silicon on any desired process node in almost push-button user friendly way. The silicon manager keeps track for the library about status of silicon implementations of the components. With this entire flow one gets not only a database of customized components but also their silicon implementation which is very helpful in architecture exploration or creation as it provides a pre-knowledge about the silicon properties of these components (like area, power, timing, configuration size etc.) to make architectural decisions. Chapter 4 will present several experiments using these tools.

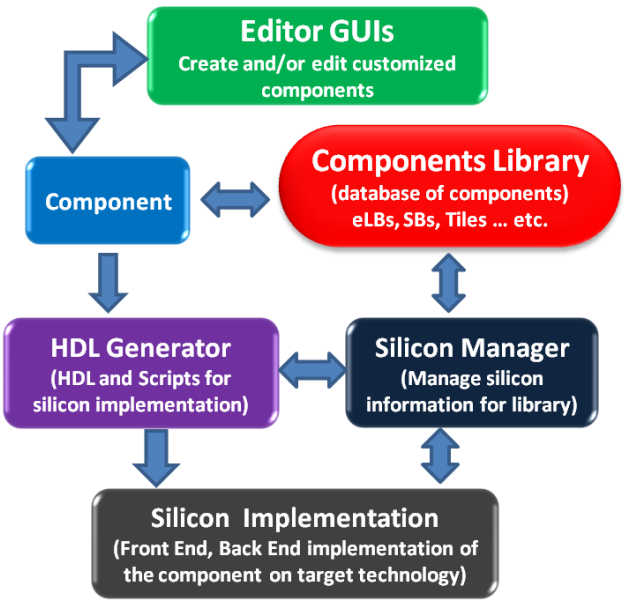


Fig. 3.7: Library Manager flow/overview

1- GUI Wizards & Editors

Figure 3.8 shows snapshot of the main library tree window (also visible in figure 3.9). It can be seen that the library contains several components (Tiles) with different characteristics. It can also be noticed there are two libraries in the library manager, in this case CTY_Phase1_Part1_wiz and CTY_Phase1_Part2 respectively. There is no limit of the size or number of libraries for the library manager. Having multiple libraries provides the obvious benefits of categorizing components for ease of use and management, e.g. one library can have components more suited for higher speed and one for higher density etc. Figure 3.9 illustrates some portions of the component creation/edition GUIs. The current infrastructure provides VPR like "what-if" architectural experiments ease infrastructure at a graphical level which in some aspects is stronger than it and in some aspects weaker at present because it is still under investigation to find the most appropriate ways.

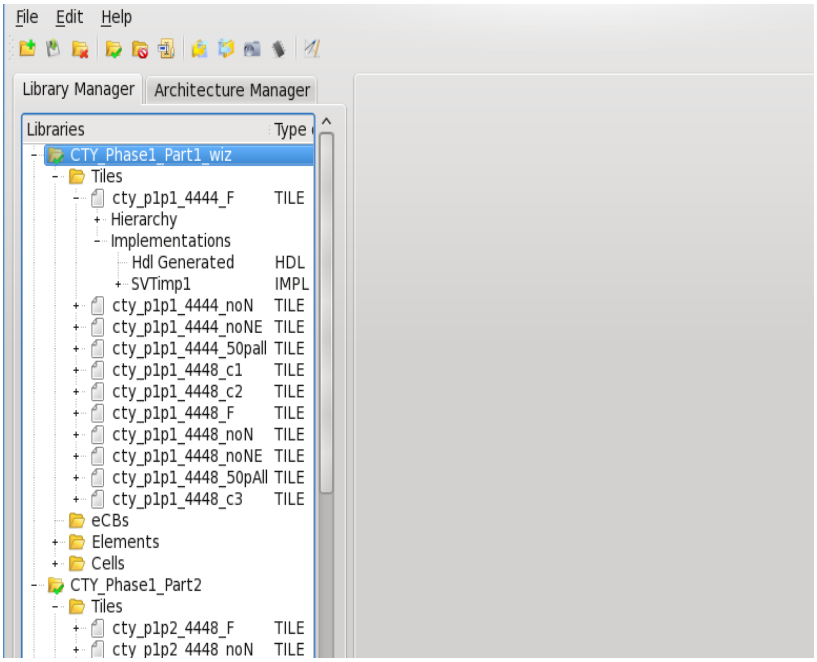


Fig. 3.8: Library Manager: library tree, components explorer

The GUIs provide a hybrid mix of automatic generation and fine grain customization of architecture parameters. Top of figure 3.9 shows the main window for creating a component, the fundamental parameters like LUT size, cluster size, channel size can be provided here and if desired a component based on default architecture model can instantly be created in a single click. If desired the fine grain tuning of architecture parameters (similar like fc,in and fc,out of scientific literature, figure 3.3) can be customized with the help of user friendly GUIs as shown in figure. This at present is more time consuming than VPR but provides greater degree of "what-if" freedom in comparison. Furthermore since everything is GUI based it highly accelerates the creation time (minutes instead of hours) compared to past manual entry styles which VPR improved with high level architecture description [Betz&Rose00, 4.11]. In VPR the enormous ease comes by specifying some key parameters in architecture file and then it automatically translates that into complete architecture with automatically resolving some architectural conflicts by the underlying algorithms [Betz&Rose00, 4.11]. This while gives great freedom to FPGA architect but at the same time slightly looses the "what-if" motivation. As it is the algorithm which decides what to do, so if one wants to do something different the software must be modified, furthermore the VPR model takes high benefit of the logical equivalence of the logic blocks due to fully connected local interconnect, both for CAD flow and target hardware as it facilitates many things but at the same time almost restrict the experiments conducted with the local Interconnect connection box model. Since most of the research works done in last decade are fully or partially based on VPR, in many cases they have conducted similar

kinds of experiments which were addressed in [Betz&Rose99, 4.1]. [Lemieux&Lewis01, 4.24] addressed the issue of fully connected local interconnect and explored reducing the flexibility (hence area overhead due to it) without degrading critical path. Chapter 4 will discuss experiments to investigate "what-if" local interconnect is completely removed scenario in terms of area, power, speed and keeping the consideration of soft eFPGA issues.

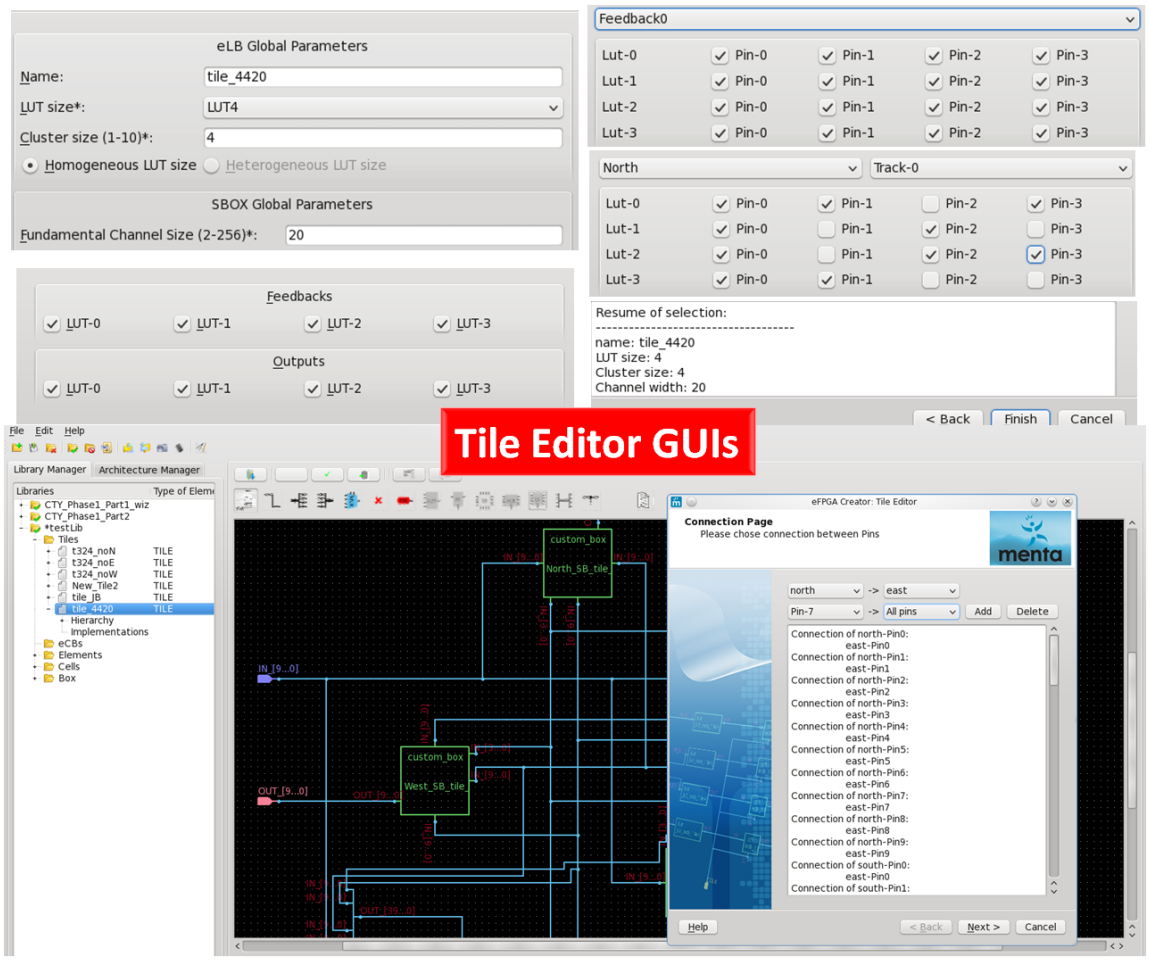


Fig. 3.9: Library Manager: some snapshots of editor GUIs

2- Hardware Generation

Library manager has built-in hardware generation. This allows instantly creating VHDL of any component in the library. As the eFPGA is pure soft, the generated HDL is technology independent, so it can be implemented on any process node using the standard cell libraries of that process. To further facilitate the implementation the hardware generator automatically creates the basic front-end and back-end scripts for the chosen process (ST65nm in case of this thesis) for the synthesis and place & route tools of Synopsys and Cadence. This highly accelerates and facilitates the implementation of the design by saving all the time to manually write these scripts and edit them for minor changes. All is automatically done by the help of graphical tools shown in figure 3.10.

In addition to that, the soft target independent nature of eFPGA and full support of hardware and script generation allows creating multiple implementations of a same component. They can be on different nodes or with different options on same node, e.g. GP (General Purpose)/LP (Low Power) or different threshold voltages etc. This highly facilitates the decision making in architecture generation where one can readily see silicon tradeoffs depending on target needs. The silicon manager (figure 3.7) manages the information for correspondence with the library components. This also helps to reuse the silicon implementations since

one only have to implement them once. Furthermore based on the maturity status of these implementations a better informed decision can be made at architecture level. [S-6] presented the concept as a seven stages (7even Star) evaluation method where a more confident decision of choice of components can be made based on their silicon maturity which is crucial in commercial scenario where silicon proof is very important (getting more and more crucial in beyond 90nm technologies). This further highlights the motivations for the two flows of eFPGA Creator (Creation and Exploration) that were discussed above.

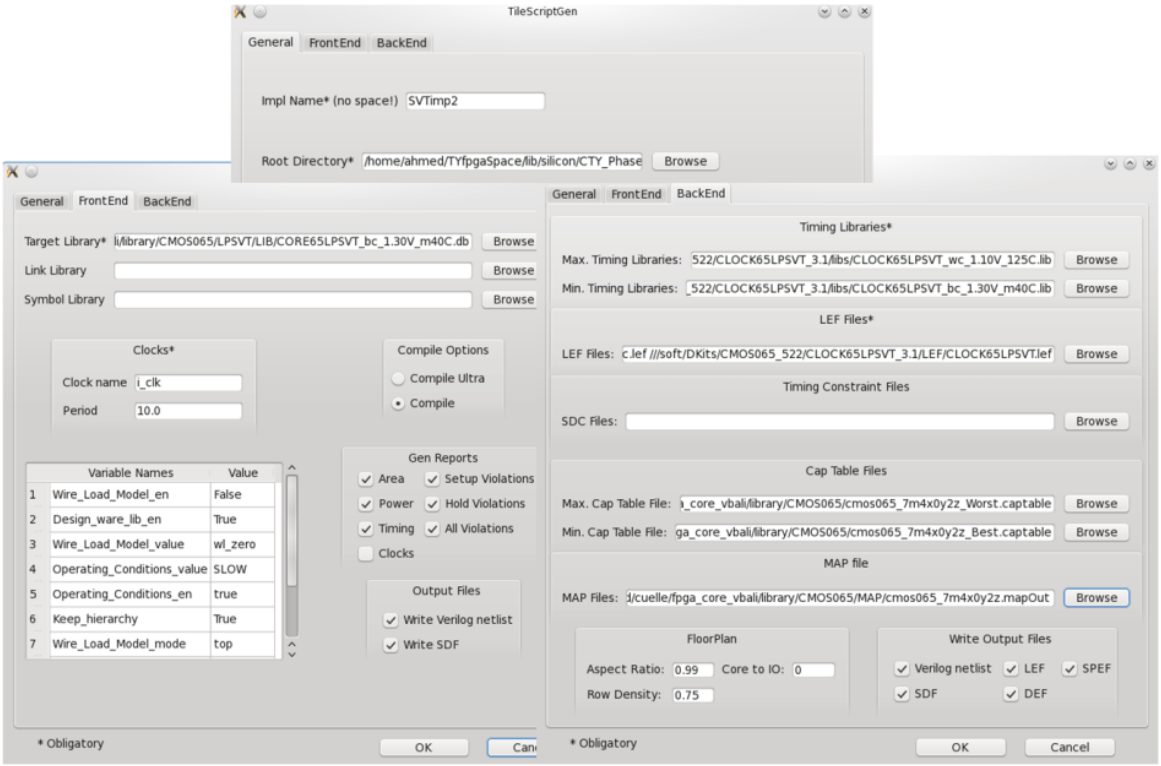


Fig. 3.10: Library Manager: Scripts generator

3.3.3 Architecture Manager (eFPGA Core creation)

The architecture manager provides an infrastructure of creating eFPGA Core using the components from library manager. Figure 3.11 presents the flow of architecture manager (notice it is tightly coupled with library manager as they both share information), it is briefly discussed below and then main components are discussed with snapshots of some portions of the tool (like library manager was discussed).

The flow starts with the GUI editor tools which help to create a new architecture or edit an existing one. All the components used inside the architecture are obtained from library manager. It automatically creates missing or required new components for core generation (e.g. components at the boundary of core, I/Os etc.) if they are not already available in library and store them in the library for future reuse. Similar to library manager the architecture manager has hardware generator which creates HDL of the Core and of the components inside the core (if the HDL was not already generated for them and put it in the library manager). Scripts are generated for core implementation with the options to choose which implementation of the components one want to use (theme that was discussed above in library manager). The core silicon manager keeps track of silicon implementations (in analogy to silicon manager of library manager). Finally the architecture files generator creates all the architecture files needed by eFPGA Programmer to map benchmark applications on the created architecture. Main components of architecture manager are described below.

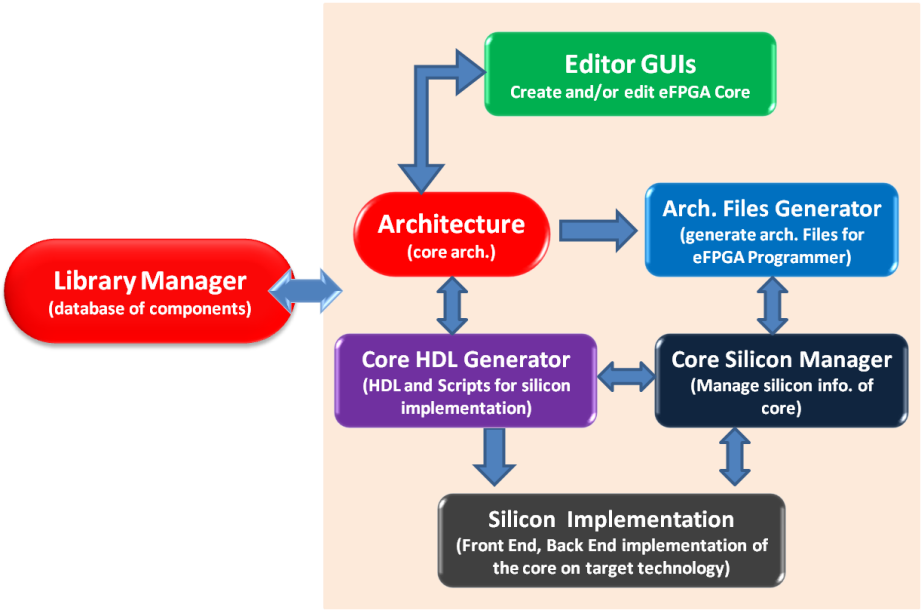


Fig. 3.11: Architecture Manager flow/overview

1- GUI wizards & Editors

Figure 3.12 shows snapshot of the main tree window of architecture manager (also visible in figure 3.13). It lists all the created architectures and provides infrastructure to create, edit, implement and analyze (discussed in next section) architectures. Figure 3.13 shows the main windows for creating a new architecture. It can be seen how simple and straightforward it is and the tight integration with library manager. A main tile is selected from a selected library; the architecture parameters (number of LUTs, I/Os) are provided either in form of quantity or parameters, and if no further customization is desired, instantly the architecture is created and added to architecture manager. The diagram on bottom shows graphical editor of architecture which allows deeper tuning of architecture like fine tuning the I/O requirements, architectural heterogeneity etc. which is lot easier and straight forward to graphically manipulate and edit compared to text entries or graphical wizards.

At this point one can further visualize the benefits of the library of components. The architecture manager uses only components from library and since it is tightly coupled with library manager (figure 3.11) if some missing component is found due to some customizations decided (like components on border, specialized I/O etc.) it is automatically created by architecture manager through library manager and added to library (including the option of finer customization using library manager component editors). Thus high reuse is obtained due to library based structure.

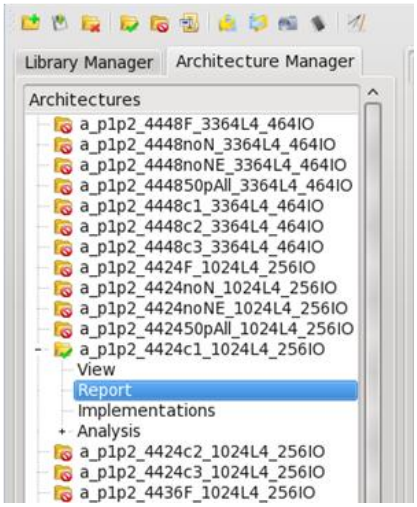


Fig. 3.12: Architecture manager

2- Hardware Generation

In a similar fashion like library manager there is a hardware generator in architecture manager which generates hardware for the core. However there is a difference between the two as the implementation of core is not simple like a component. The library based approach further helps here as the hardware of the components is already implemented in the library manager flow. The core generator reuses the implemented design of components (.lib, .lef etc. files if already exist) to build up the core. The core generator creates the top level VHDL which wraps all the building components which are already implemented and available in library. Hence the implementation is/can be order of magnitude easier and faster compared to starting from the scratch. If some new components arise in creation of core they will be implemented using the library manager flow and will be available for future reuse. Furthermore since all the building components are already implemented, it further helps to have rough pre-idea of silicon properties of architecture (like area, static power, silicon status etc.) without even implementing it which further accelerates the design space exploration.

3- Architecture files generation

Architecture files generator creates all the architecture files needed for eFPGA Programmer (figure 3.5) to implement benchmark applications on the created architecture. From figure 3.11 it can be seen that the tool gathers all the architectural and silicon information and builds the detailed architecture files of the created architecture. This process is long and computation intensive as architecture files are very large in size (also proportional with architecture size) as they carry all the routing architecture, timing information etc. Figure 3.14 shows snapshot of the generator tool. To facilitate computation efficiency exploiting the multi-core PCs which are common now, the tool provides option of multithreading and compute in parallel. In the figure the computation is in progress on four threads. This helps accelerating the generation time.

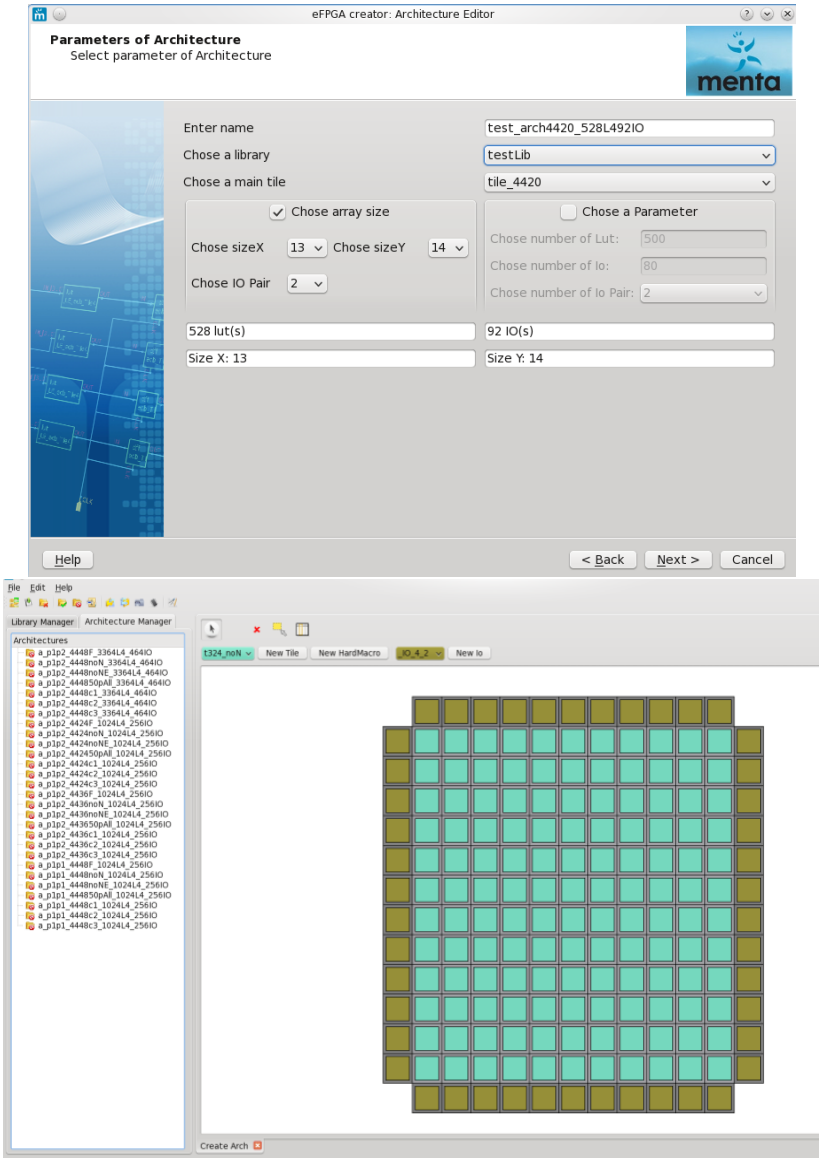


Fig. 3.13: eFPGA Creator Architecture Manager Snapshots

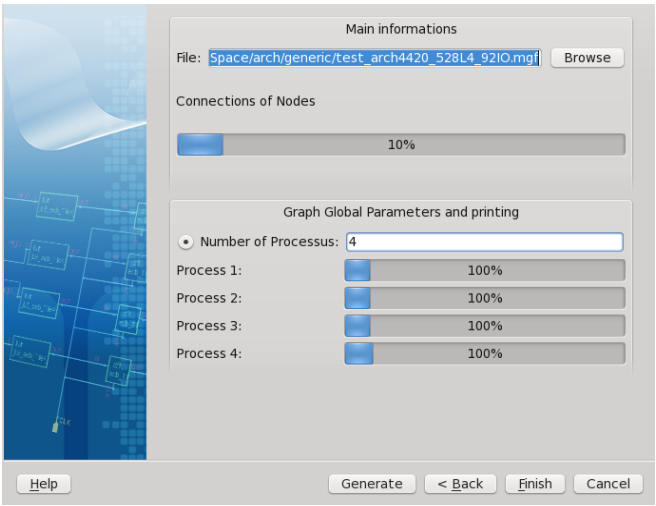


Fig. 3.14: eFPGA Creator Arch. Files generator

3.3.4 Analyzer (Design Space exploration)

The analyzer infrastructure facilitates analysis of benchmark applications mapped on the architecture. To make analysis more effective and illustrative, support for GUI is used. The early inspirations for creating graphical tools for analysis evolved from the work in [S-9]. With the rich graphical infrastructure of Qt it became easier and effective to upgrade some of the analysis tools of [S-9] and create several new tools in addition which became possible due to advanced graphics facilities of Qt and their integration into eFPGA Creator. A brief description of the flow is presented below and then main elements that are analyzed are discussed in more detail.

Figure 3.15 shows the flow of analyzer, the theme diagram of eFPGA creator (figure 3.6) is also represented on bottom right to facilitate discussions. It is interesting to note that the analyzer is closely coupled with the library and architecture managers. The flow of analyzer is simple and as follows. The application benchmarks are implemented on architecture or a group of architectures from the architecture manager database. The results of implementations are analyzed using analyzer tools suite; based on them the appropriate architecture that closely matches the requirements is selected. The analysis also helps to find the fine tuning direction for customization. Based on the clues obtained from analysis one can either go to the architecture manager to create a new architecture/new implementation of architecture (e.g. implementation with different threshold voltage for power vs speed tradeoffs etc.), or go to the library manager to create a new component which is more appropriate for target application (e.g. routing characteristics, LUT/cluster size, hard components etc.). Hence it can be noted that all the tools are closely interlinked with each other.

All the analysis tools are integrated in the eFPGA Creator making it convenient and faster to perform the flow of figure 3.15. Figure 3.16 shows a global view of how the analysis tools are closely coupled inside the eFPGA creator. Brief details of some major areas of analysis are discussed below.

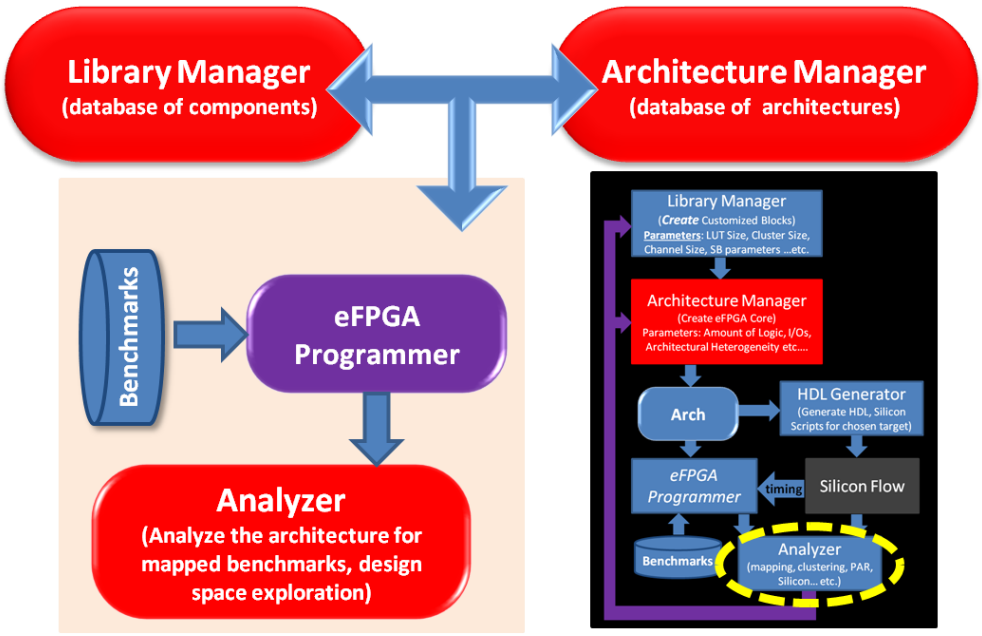


Fig. 3.15: eFPGA Creator Analyzer

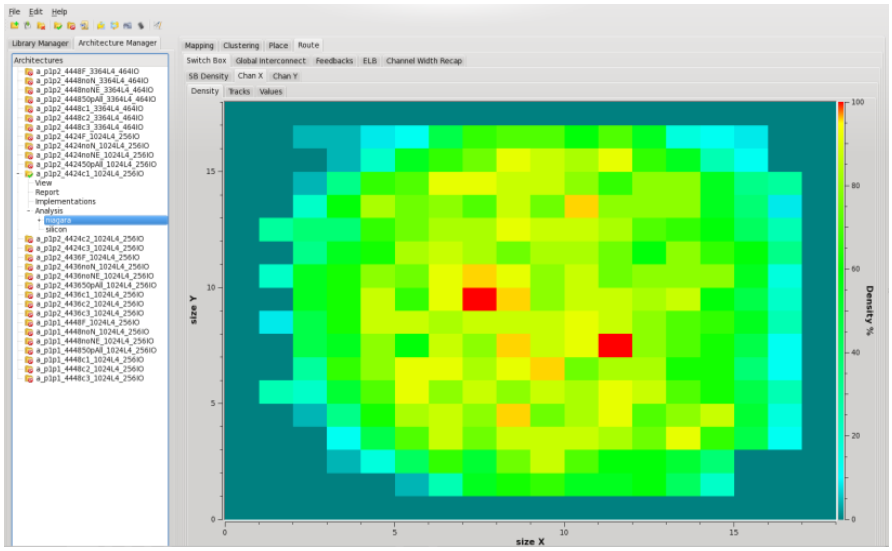


Fig. 3.16: eFPGA Creator Analyzer GUI cockpit

1- Mapping, Clustering & Placement

Mapping analyzer tool is shown in figure 3.17. It helps to analyze the LUT mapping statistics of the mapped application. Chapter 2.1.2 discussed about the LUT mapping efficiency issue for different LUT size in the state of art discussion and noted that the efficiency of mapping was different for different LUT sizes and goes down as LUT size increases. This tool helps to visualize these issues for experiments. In figure 3.17 the tool is depicting statistics of a mapped application on a LUT4 based architecture and it can be observed that for this application almost half of all the LUTs used are fully utilized as LUT4.

Clustering analyzer tool is depicted in figure 3.18. It helps to analyze how efficiently the LUTs are utilized in cluster (eLB), e.g. if eLB has a cluster size of 4 and there are 100 eLBs used by the application the tool will analyze how efficiently they were packed up with the LUTs inside them. This information can be easily represented in a table like shown in figure 3.17 for LUT mapping but thanks to the graphical infrastructure the tool helps not only to see the efficiency but also the visual positions of the distribution inside the core. In figure 3.18 it can be seen that almost all the clusters are fully (100%) packed for this application and those that are not are clearly identified.

Placement analyzer tool is shown in figure 3.19; it is similar in principle like the clustering analyzer tool and provides a visual view of all the components placed/used in the architecture including the I/Os.

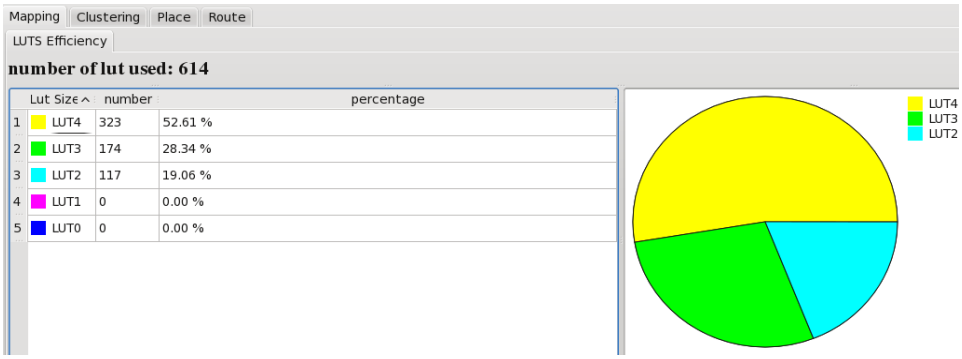


Fig. 3.17: LUT mapping analyzer

Fig. 3.18: Clustering efficiency analyzer

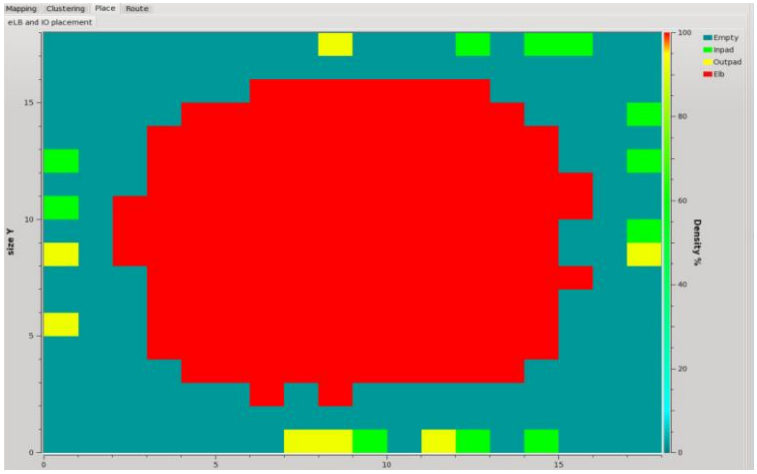


Fig. 3.19: Placement analyzer

2- SB & eLB

Routing is the most complex and exciting part of FPGA architecture research. Numerous tools were created in this thesis work for analyzing routing or matters closely linked with routing. Figure 3.20 shows the traffic analysis of the eLB in connection with SB (figure 3.2). On the top of the figure some of the routing analyses are visible and one can get a feeling how in-depth the routing is analyzed. The analysis details for a mapped application are depicted in a table and sowing how the incoming traffic of eLB is distributed from outside world point of view. The first column shows the inputs of eLB and for each of that the possible traffic from four dimensions/ports of SB (North, South, East and West) and from the feedbacks of eLB itself (represented as South West based on figure 3.2) is analyzed. Such analyses greatly help in architectural exploration for customizing the switch box architecture which will be discussed in detail in chapter 4.

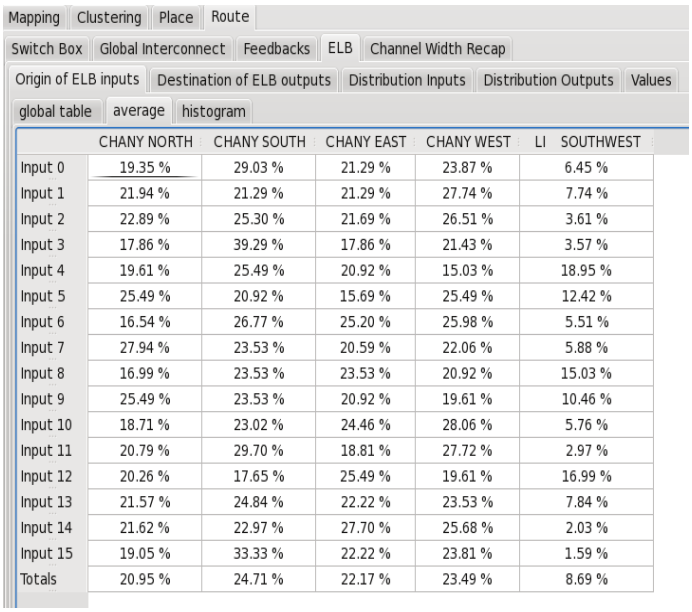
3- Global Routing

The global routing analysis provides high level architecture overview to investigate issues like channel width requirements, routing congestions, hop analysis etc.

Channel analysis

Figure 3.21 shows the snapshots of the tools for channel analysis and routing congestion of a mapped application. The combination of tabular and graphical information helps to analyze key information which is crucial in architectural considerations. In figure 3.21 the routing statistics and corresponding congestion plot is illustrated. It can be seen that the application took maximum of 24 tracks to successfully route on the target architecture which in this particular case has physical channel size of 24 also, which means the architecture is utilized to its maximum limits (most of the time it is not the case), what is further interesting to note is that this high value of 24 tracks was only used 3 times during Place and Route, two of them can be seen in congestion graph (showing routing congestion for horizontal channels ChanX, the third one is

on ChanY). On average the channel utilization remains at around 15 as shown in the table, meaning almost half of the tracks are never used. This shows how complex the routing challenges are. These issues will be discussed in detail in chapter 4. It can be observed how much research ease is provided by these graphical tools for FPGA architect.



	CHANY NORTH	CHANY SOUTH	CHANY EAST	CHANY WEST	LI	SOUTHWEST
Input 0	19.35 %	29.03 %	21.29 %	23.87 %		6.45 %
Input 1	21.94 %	21.29 %	21.29 %	27.74 %		7.74 %
Input 2	22.89 %	25.30 %	21.69 %	26.51 %		3.61 %
Input 3	17.86 %	39.29 %	17.86 %	21.43 %		3.57 %
Input 4	19.61 %	25.49 %	20.92 %	15.03 %		18.95 %
Input 5	25.49 %	20.92 %	15.69 %	25.49 %		12.42 %
Input 6	16.54 %	26.77 %	25.20 %	25.98 %		5.51 %
Input 7	27.94 %	23.53 %	20.59 %	22.06 %		5.88 %
Input 8	16.99 %	23.53 %	23.53 %	20.92 %		15.03 %
Input 9	25.49 %	23.53 %	20.92 %	19.61 %		10.46 %
Input 10	18.71 %	23.02 %	24.46 %	28.06 %		5.76 %
Input 11	20.79 %	29.70 %	18.81 %	27.72 %		2.97 %
Input 12	20.26 %	17.65 %	25.49 %	19.61 %		16.99 %
Input 13	21.57 %	24.84 %	22.22 %	23.53 %		7.84 %
Input 14	21.62 %	22.97 %	27.70 %	25.68 %		2.03 %
Input 15	19.05 %	33.33 %	22.22 %	23.81 %		1.59 %
Totals	20.95 %	24.71 %	22.17 %	23.49 %		8.69 %

Fig. 3.20: SB-eLB traffic analyzer

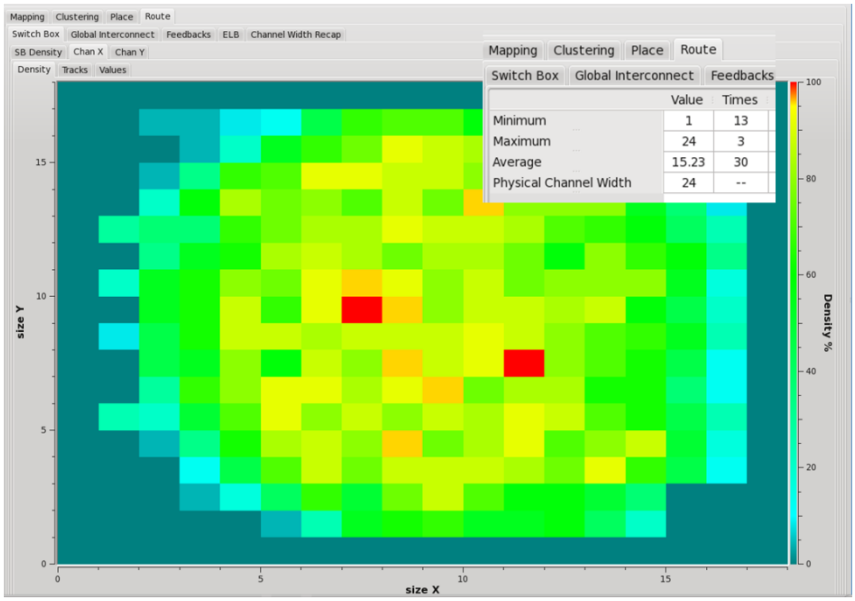


Fig. 3.21: Global routing analysis

HOP Analysis

Another important aspect in global routing is HOP as was discussed in chapter 2.1.2. It helps analyzing the global traffic in the architecture and investigating the exploration of long wires. Figure 3.22 shows the snapshot of hop analyzer which allows investigating the hop analysis of benchmark applications mapped on the architecture. Chapter 4 will discuss hop analysis experiments in more detail.

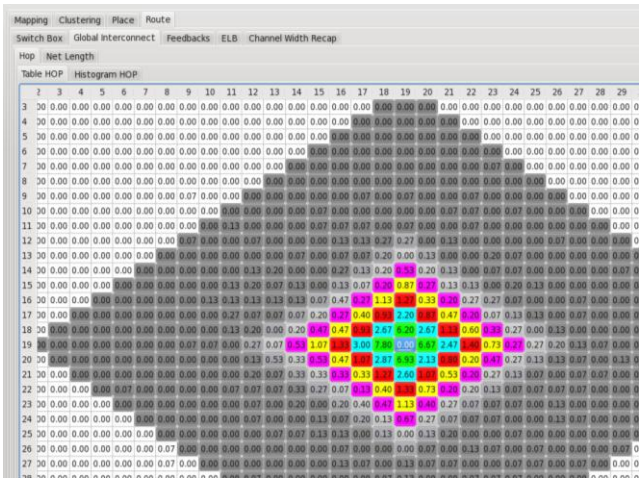


Fig. 3.22: Hop Analysis

4- Silicon

The silicon analyses are slightly different than the architectural analyses discussed above in a sense that they provide fixed information which is benchmark independent. The elements of silicon analyses are aspects like area, power (static), timing, configuration size etc. of components of eFPGA and of the core. Chapter 4 will discuss some silicon analysis tools which were made for early experiments using TCL/Tk [S-9], some of them are ported to eFPGA Creator analyzer. Figure 3.23 shows the area distribution analysis tool which shows the breakdown of main components of the tile. An interesting point to observe in figure 3.23 is that due to soft implementation the contribution ratio between logic and configuration is slightly different than the classical layout based FPGA designs with pass-transistor routing etc. This is due to the fact that soft eFPGA uses multiplexers for all kind of switching in the architecture hence the physical hardware is relatively logic dominant compared to configuration as the amount of configuration bits in a multiplexer based routing compared to pass-transistor based routing is less due to obvious reasons (with a tradeoff of multiplexer vs pass-transistor area). This also brings some interesting considerations while exploring the architecture to keep the perspective of soft eFPGA which leads to multiplexer based routing where configuration size/bits can abruptly change with slight change in parameters if not carefully considered; chapter 4 will address these aspects further.

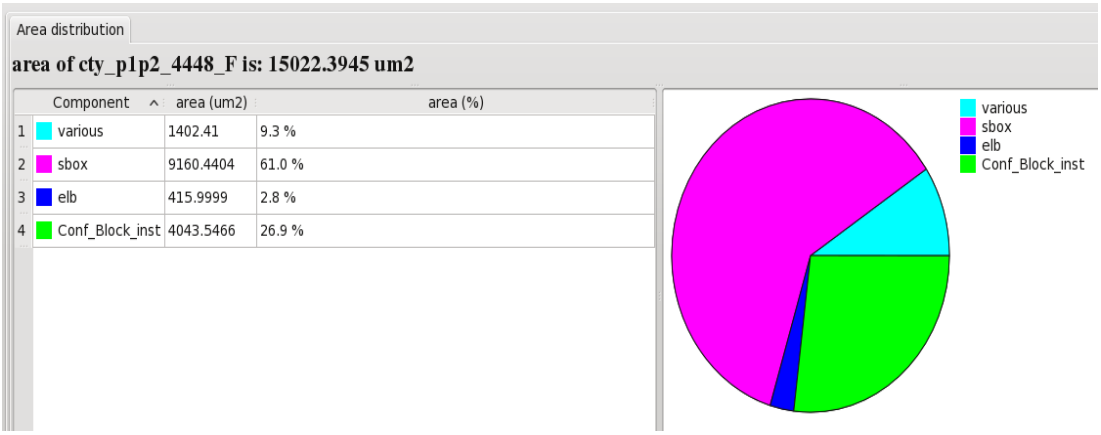


Fig. 3.23: Tile Area distribution analysis

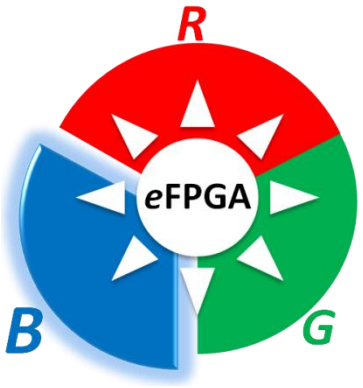
3.4 Summary

This chapter discussed fundamentals of eFPGA hardware and CAD (eFPGA Programmer) tools. It provided an overview of the eFPGA Creator tools suite; explaining its fundamentals, details of building blocks, differentiations/comparison of the tools with comparative/existing approaches (e.g. VPR etc.), contributions of this thesis work. The key points observed/discussed are described below.

- eFPGA under experimentation has classical FPGA-like island-style architecture, it uses uni-directional (single driver) routing architecture and it has no classical connection blocks (CB) or local interconnect (LI). A unified switch block (SB) performs all switching. It has highly scalable blocks based structure
- eFPGA is pure soft (written in VHDL) and technology independent
- eFPGA Programmer provides standard RTL programming flow for eFPGA
- eFPGA Creator tools suite provides a user friendly GUI based infrastructure for creation and exploration of customized eFPGAs. The infrastructure is composed of several building blocks (library manager, architecture manager, hardware generator, analyzers etc.). Library manager provides an infrastructure to create customized blocks/components (eLBs, SBs, Tiles etc.) and also serves as a database of components. Architecture manager helps create eFPGA core and also acts as database of architectures. Hardware generators allow automatic hardware and scripts generation of created components/architectures for silicon implementation. Analyzers help to explore the architectures.

Next chapters address the exploration of eFPGA architecture and investigation of eFPGAs in systems using these tools and knowledge of chapter 2. Chapter 4 will address in detail the architectural exploration of eFPGA to investigate creation of efficient eFPGAs. Chapter 5 will discuss integration of eFPGAs in SoCs scenario, reconfigurable acceleration and some beyond classics perspectives.

Chapter 4: eFPGA Architectural Explorations [B]



The FPGA architecture research in many aspects is similar to the famous quote of Antoine de Saint-Exupéry:

"*Perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away*". (wikiquote)

This chapter presents in detail the eFPGA architectural exploration research contributions of this thesis work. Since eFPGA has a classical LUT based FPGA-like architecture, the research conducted is similar like FPGA architecture research addressed in scientific literature (term eFPGAs and FPGAs will be often synonymously used in this chapter). However issues related with soft eFPGAs have been kept in consideration as a motivation during research explorations. This chapter discusses experiments conducted in the beginning and at the end of the thesis work. It will provide some insight among connecting motivations of different dimensions of thesis work. The outline of the chapter is as follows.

Section 4.1 explains in detail the basic explorations done on almost all fundamental aspects of FPGA architecture research for general overview based on the motivations obtained from state of the art and academic research (chapter 2). This section explores fundamental issues like LUT size, cluster size, channel analysis, routing challenges, issues of power consumption in beyond 90nm nodes and basic comparisons from state of the art solutions. Most of the work addressed in this section is based on [S-9]. The explorations addressed in this section provided some of the motivations behind the creation of eFPGA Creator, eFPGA Programmer and served as a foundation start point of system level research exploration (will be discussed in chapter 5) by providing the flexible hardware of eFPGAs for experimentation with silicon information [S-7][S-8].

Section 4.2 addresses advanced experiments of customizing eFPGAs using knowledge of section 4.1 and exploration ease and infrastructure provided by eFPGA Creator. This section presents detailed experiments for analyzing tradeoffs in area, power and speed on ST65nm process by exploring different architectural customization for creation of customized eFPGAs which are validated by mapping benchmark applications to judge their routability tradeoffs in response of customizations done. This work provides more silicon efficient architectures with higher customization compared to basic customizations of section 4.1.

4.1 Basic Explorations for General Overview

This section discusses in detail the basic eFPGA architectural explorations with eFPGA CAD and silicon analysis to obtain a general overview of challenges and research directions. Most of the work presented in this section is based on [S-9]. This section first briefly presents the experimentation methodology by presenting the experimentation flow, CAD tools, benchmark applications, adaptable VHDL of eFPGA hardware and exploration GUI tools for architectural and silicon analysis. Then later sub sections present in detail the explorations for LUT size, LUT mapping efficiencies vs LUT size, clustering and channel, routing challenges and investigations, challenges of power consumption in beyond 90nm nodes and some basic comparison of results with state of the art to get inspiration for research challenges and directions.

4.1.1 Experimentation Methodology

This section presents the experimentation methodology for the explorations addressed in section 4.1. It presents the experimentation flow, CAD flow for mapping benchmark applications on different eFPGA architectures, the adaptable VHDL of eFPGA hardware to facilitate changing several fundamental parameters of eFPGA and visualizing the silicon properties by silicon implementation on different target nodes (90nm, 65nm and 45nm). To facilitate the analysis several basic GUI tools using Tcl/Tk, HTML were created which are briefly discussed.

1- Experimentation flow and Benchmarks

Figure 4.1a outlines the global flow used for eFPGA architectural explorations in section 4.1. It can be observed that the flow performs the exploration of fundamental parameters of eFPGA (LUT size, cluster size and channel size). The flow is composed of three main components; eFPGA CAD, eFPGA adaptable VHDL and Analysis tools. The flow and components are briefly outlined below.

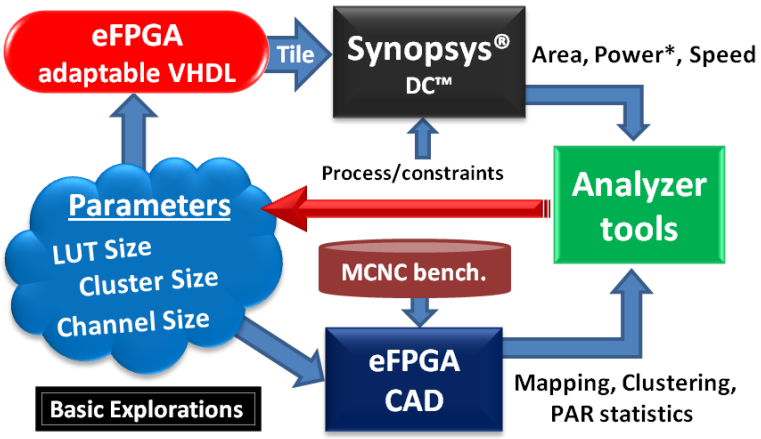


Fig 4.1a: Experimentation flow for basic explorations

eFPGA CAD: These early experiments [S-9] used the VPR (version 4.3) tools of Toronto University [Betz&Rose99, 4.1] for the CAD flow for mapping the bench marks. They were modified slightly for the experiments (unidirectional architecture). These tools were selected as their quality is well recognized in academics and industry with high citations. Furthermore they target mesh based architecture which was target of this work also so explorations done with these experiments were useful with later tools (eFPGA Programmer, eFPGA Creator) which were under construction and investigation in parallel while these experiments were conducted. The simplicity and limitations of the basic flow can be seen in figure 4.1a. The eFPGA CAD allows mapping benchmark applications on different eFPGA architectures and effect of changing the fundamental parameters can be explored and visualized by analysis tools. However it is important to note that in the basic flow (figure 4.1a) there is no link between eFPGA CAD and silicon information, due to that comparison of different architectures in terms of speed for the mapped benchmark applications is not possible in the basic exploration. eFPGA Creator (chapter 3) enabled advanced explorations which will be addressed in section 4.2. The benchmark applications used for the experiments are outlined below.

Benchmarks: For the benchmarks, the MCNC benchmarks were used [Yang91, 4.82]. We selected the 20 MCNC benchmarks (Toronto 20, [Betz&Rose99, 4.1][Njuguna&Jain08, 4.81]) which are common in the scientific literature related to FPGA architecture, making it easier for researchers to compare results of different research works. Table 4.1 lists the circuits with their I/O characteristics and number of LUT4 (flip-flops are not shown) required to map these circuits using the SIS tools of Berkeley University [4.83]. It is interesting to note that the benchmark circuits cover a wide variety of circuits, like high I/O, high logic etc., later sections will analyze some findings about channel based on these diversities.

Circuit	Inputs	Outputs	No. of LUT-4
ex5p	8	63	1064
tseng	52	122	1047
apex4	9	19	1262
misex3	14	14	1397
alu4	14	8	1522
diffeq	64	39	1497
dsip	229	197	1370
seq	41	35	1750
apex2	38	3	1878
des	256	245	1591
bigkey	229	197	1707
s298	4	6	1931
spla	16	46	3690
frisc	20	116	3556
elliptic	131	114	3604
pdc	16	40	4575
ex1010	10	10	4598
s38417	29	106	6406
s38584	38	304	6447
clma	62	82	8383

Table 4.1: 20 MCNC benchmarks, statistics with LUT4 mapping using SIS

2- Adaptable HDL of eFPGA: Silicon exploration ease

As the main motivation since beginning was towards a soft eFPGA, it was essential to have a highly adaptable and flexible HDL of eFPGA which can be customized for different fundamental architectural parameters (LUT size, cluster size, channel size etc.). This aspect, as was discussed in chapter 3 (with pros and cons) in some ways differentiates this work compared to several other FPGA architecture exploration works found in scientific literature (most notable VPR/VPR-like).

To facilitate experimentations a highly generic architecture of eFPGA was created in VHDL which is used in all these basic explorations presented in sections below. The design is highly flexible and with the generics of VHDL several fundamental parameters like LUT size, cluster size, channel size, array dimensions, even multi-context (not discussed in this work) etc. can be controlled. This allows rapid silicon implementation of explored parameters along with CAD experimentation of mapping benchmark applications on the explored architectures as illustrated in figure 4.1a. Figure 4.1 illustrates the architectural fundamentals of eFPGA tile (chapter 3). The adaptable VHDL of eFPGA allows experimenting different architecture possibilities by changing all the fundamental parameters as shown in figure. This section addresses investigations based on adaptable VHDL of eFPGA. The SB topology used throughout the experiments is Disjoint [Wilton97, 4.25][Masud99, 4.35]. This adaptable VHDL of eFPGA provided key motivations for HDL generator of eFPGA Creator (chapter 3), which will be used for advanced explorations in section 4.2 allowing customizations of SB-R and SB-eLB multiplexers.

Silicon implementation: For the silicon libraries this work used 65nm LP, 90nm GP and 45nm LS process technologies of ST provided by CMP (cmp.imag.fr). Except few comparative cases all work in this thesis is on 65nmLP for which the widest range of libraries was available for analyzing different parameters like effect of threshold voltage, temperature, supply voltage etc. Figure 4.1a shows the flow. The VHDL of tile customized to explored parameters is synthesized (front-end) with Synopsys Design Compiler on the chosen process node parameters and provides good approximate silicon statistics of area, power* (static, dynamic on statistical toggle rates; section 4.1.6 will address this methodology as pessimistic power) and speed for the tile (not the benchmark execution speed!, section 4.2 will address benchmark speed).

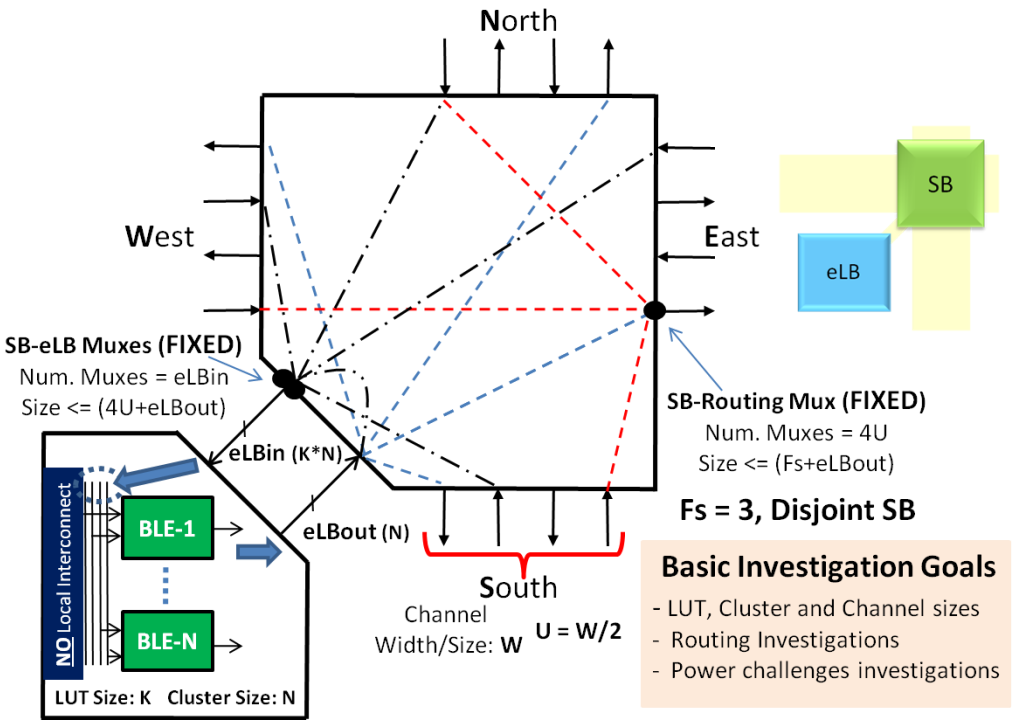


Fig 4.1: Basic Hardware (Tile) of eFPGA Architecture

3- Basic GUI exploration tools

In many ways the FPGA architecture research revolves around tools or perhaps is all about tools; tools for design, tools for implementation, tools for exploration and even tools to explore exploration. To facilitate the analysis of large amount of exploration data both architectural (FPGA CAD mapping, PAR etc.) and Silicon (Area, Power, Timing etc.) several basic GUI tools were created. These tools and their concepts later became the foundation of high-end GUI Analyzer tools of eFPGA Creator which were discussed in chapter 3. The created tools can be divided into two main categories, architectural exploration tools and silicon exploration tools. The deeper details about these tools, how they work and made is proprietary of Menta and is beyond scope of discussions, the thesis only presents and discusses the results obtained with the help of these tools.

Architectural Exploration tools analyze the outcomes of the application mappings by the CAD tools on the explored architectures as shown in figure 4.1a. The analysis statistics includes elements like LUT mapping, clustering, routing, hop etc. Several tools with HTML based GUI for elaborated outputs were created, the major tools included: Mapping Analyzer, Clustering Analyzer, Routing Analyzer, Channel Analyzer and Hop Analyzer.

Silicon Exploration tools help graphically analyzing the outputs of silicon implementation for area, power and speed for different architectures under exploration as shown in figure 4.1a. They highly facilitate cross analysis of different architectures and visualize the silicon tradeoffs to help directing the customization direction in co-observation with the Architectural Exploration tools. These tools were created with the help of Tcl/Tk (www.tcl.tk). Creating the tools using Tcl/Tk while giving nice features of creating basic graphics (Tk GUI) in addition also helped in the fact that Tcl (Tool Command Language) is often used like a de-facto standard for communicating with silicon tools. This also provided a dual help; easy to learn and creating scripts for silicon tools.

Next sub sections of section 4.1 present results of exploring several fundamental architectural parameters using the methodology described above, majority of results presented are from [S-9].

4.1.2 LUT Size

The first and foremost element and workhorse of an FPGA is a Look-Up Table (LUT). This section presents the exploration done for LUT size. Several results obtained were similar like addressed by previous works without much surprise as clues were already obtained from works like [Ahmed&Rose04, 4.8][Betz&Rose99, 4.1][Altera07, 1.18]. However doing these experiments again was interesting because it was essential to get silicon and architectural statistics for under exploration eFPGA architecture and while doing so the thesis work did some complimentary work also to further examine some issues. The details of explorations are described below.

1- Mapping efficiency vs LUT Size

Bigger LUT sizes have relatively poor mapping efficiency (from logic packing ability of CAD standpoint) compared to smaller LUT sizes. As the LUT size grows exponentially by increasing its inputs due to obvious reasons (e.g. LUT6 is more than 4 times bigger than LUT4), the mapping efficiency is an important research area and consideration regarding the LUT size. Chapter 2 presented the Adaptive Logic Modules (ALM) innovation of Altera which finds its key motivations from mapping efficiency issues. [Altera_Lewis05, 1.22] explains different directions of investigations that Altera conducted in this regard while innovating heterogeneous and adaptable ALM logic block. However another interesting point which closely attach to the story of LUT size is that the LUT size itself is dwarfed by the sheer area that the interconnect architecture takes, it will be soon observed in silicon observations. *FPGAs are masterpiece of transistor waste;* in many aspects the FPGA architects while continuously innovating to bridge this gap also take smart benefit to mange this waste to tradeoff for some other property e.g. area can often be traded for speed. For such reasons as was discussed in chapter 2, the speed and intuitively also power (from routing perspective) benefits of bigger LUT sizes often overshadow their area overhead and mapping inefficiency. Bigger LUT sizes are common in commercial solutions with LUT4 still being most common in majority of FPGAs (from counting types/companies point of view not market share which only for Xilinx is more than 50% and it now uses LUT6).

Table 4.2 presents the mapping efficiency statistics for targeting different LUT sizes (fixed homogeneous size). The results are from the average mapping (SIS) statistics of 20 MCNC benchmarks. Figure 4.2 illustrates statistics of table 4.2 in more detail with split up contribution of partially mapped LUTs. It clearly shows the drop in mapping efficiency with increase the LUT size. The graph shows similar patterns like were observed in figure 2.6 from Altera in chapter 2. The values differ (for Altera they observed efficiencies of LUT4 around 55%, LUT5 around 35% and LUT6 around 30% respectively) as they vary from benchmarks and CAD tools however the trend of weaker efficiency of bigger LUTs is visible. To further investigate this issue some cross experiments were conducted by mapping some of the MCNC benchmarks on Xilinx Virtex-II Pro FPGA (is LUT4 based) by using Synplicity (now Synopsys) synthesizer which is well know 3rd party FPGA synthesizer for its superior quality. The results of investigation are illustrated in table 4.3. Comparing the number of LUT4 used compared to table 4.1 one can see a significant improvement. The improvement can be attributed to two main components; firstly the Virtex-II has a more complex logic block structure with heterogeneity of carry chains, muxes, multiple capabilities of LUT etc. these features indirectly decrease the raw LUT utilization, the second factor is superior algorithms of the tool compared to academic SIS tools which further optimize the consumption. Considering only the second component (for ease of discussion) the first thing that comes to mind by seeing the LUT count is that mapping algorithms would have taken more efficient use of LUT4 and contributed in decreased number of LUT in addition to better logic optimization. When we analyze the mapping efficiency in the table we see some surprising observations that mapping efficiency instead of going up (from 67.45% of SIS) has gone down to around 48% (more similar like Altera 55%). Although that does not mean an end to mapping tools which are a continued research topic and perhaps some future innovations can help better exploitation but it is apparent and illustrates that the LUT mapping efficiency phenomena is quite generic and all FPGAs (including leading) have lot of underutilized LUTs in mapped

applications. At present Altera is the only prominent state of the art example which utilizes adaptable heterogeneous LUT structures enabled by their ALM structure and corresponding CAD tools (SMT) [Altera_Lewis05, 1.22].

LUT size	Mapping Efficiency %
LUT-3	75
LUT-4	67.45
LUT-5	57
LUT-6	49.76

Table 4.2: Mapping efficiency of different LUT sizes for 20 MCNC with SIS mapping

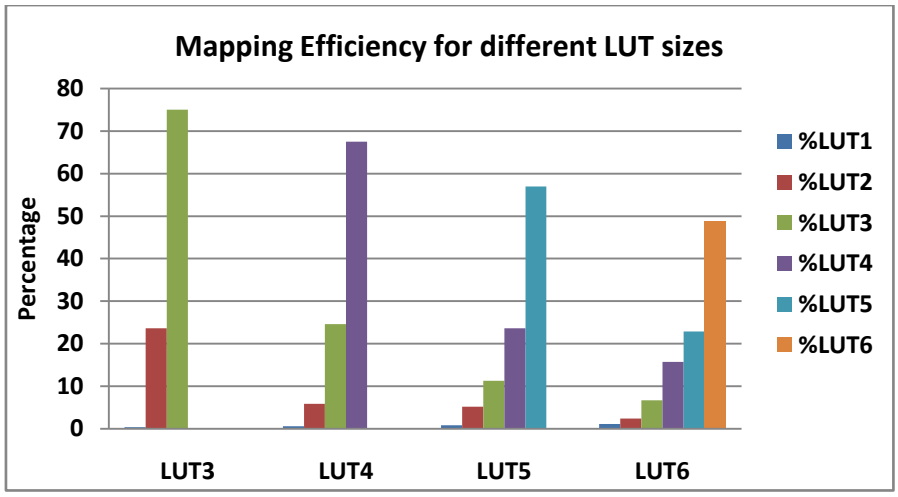


Fig. 4.2: mapping efficiency distribution for different LUT sizes for 20 MCNC with SIS mapping

Circuit	Total LUT4	%LUT-4	%LUT-3	%LUT-2
alu4	471	51.168	29.724	19.108
dsip	941	71.413	22.848	5.739
des	1149	48.651	37.076	14.273
bigkey	1542	38.132	52.399	9.468
s38417	1770	47.514	33.842	18.644
s38584	2809	43.147	41.616	15.201
clma	5134	34.496	53.428	12.076
Average		**47.8**	**38.7**	**13.5**

Table 4.3: Mapping efficiency of benchmarks on Virtex II Pro with Synplicity synthesis tool

Figure 4.3 illustrates the overview of statistics of number of LUTs needed (normalized to LUT4) for implementing 20 MCNC benchmarks while targeting different LUT sizes. We can see the effect of mapping efficiency issue by cross viewing figure 4.2. Taking as an example LUT4 (used as reference) in this case, if the benchmarks are targeted on LUT3 based architecture it requires 1.3X LUT3 compared to LUT4 instead of 2X (LUT4 has double logic capacity compared to LUT3). Going on the other side number of LUT6 (has 4 times logic capacity compared to LUT4) required to map the benchmarks are 65% (instead of 25%) compared to LUT4. The hints for these differences can be seen in figure 4.2. Since all the partially packed LUTs also occupy a complete LUT to implement them and more the LUT size increases more are partially packed LUTs e.g. In case of LUT6 we can see that more than half of LUT6 are partially packed.

Next section presents the silicon exploration of these experiments; we will keep these observations in consideration in discussions of silicon findings.

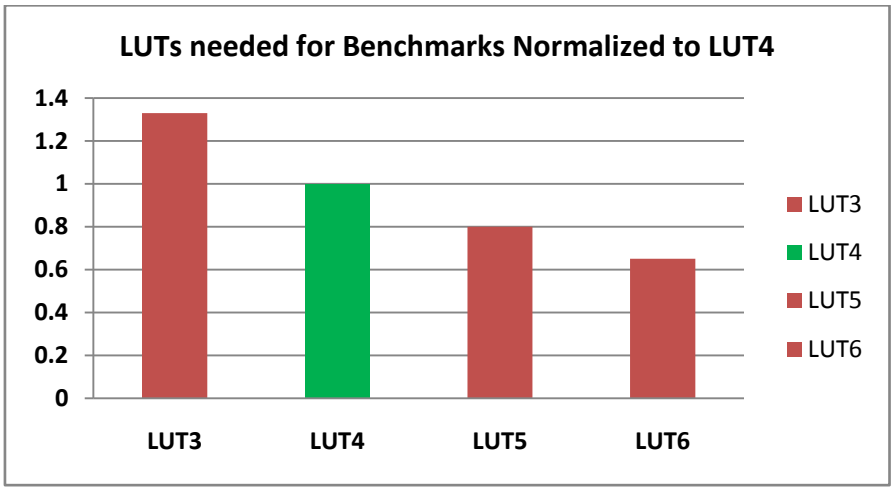


Fig. 4.3: Total LUTs needed by 20 MCNC mapping for different LUT sizes, normalized to LUT-4

2- Silicon Exploration

The above section investigated the LUT size mapping issues independent of the architecture parameters as they are independent of cluster size, actual place and route etc. In the next phase we made thorough investigation of silicon statistics based on architectural parameters which were required to effectively implement the 20 MCNC benchmarks.

We implemented the 20 MCNC benchmarks for different values of LUT size and cluster size and investigated the channel size requirements (will be discuss in more detail in next sections). Figure 4.4 illustrates silicon statistics of the tiles (the graphs are based on [S-9] with old names, Clb = eLB and Sbox+Interface = SB) of the architecture with channel size slightly higher than the average (arithmetic) channel size needed to place and route the benchmarks. The results for LUT size {3, 4, 6} and cluster size 8 are shown. We fully analyzed all the benchmarks for LUT size {4,6} (approximated for LUT3 based on observation) and found that the channel required in two cases was almost similar. This while being a good sign was bit of a surprise as what was expected is noticeably larger channel required for LUT6, since the number of inputs of eLB significantly increases when moving from LUT4 to LUT6. We can see the areas and area distributions of these architectures on 65nm LPLVT (Low Power Low Voltage Threshold), as architecture in examined case is homogeneous a tile can directly give good enough idea of architecture properties. As these silicon synthesis results are for actual hardware on a real process with architecture parameters based on suitable parameters obtained from CAD analysis of benchmarks we can make some interesting observations and investigations from them. Table 4.4 summarizes some key parameters from figure 4.4 silicon observations and figure 4.3 mapping observations. First if we observe the percentage of Tile (configuration + logic) we see that how small LUTs (eLB) in general is compared to interconnect architecture (often known to be more than 80% in general). Now as we increase the LUT size we observe the percentage of LUT in tile increases, thanks to the channel requirements observations as we discussed above. This on first impression gives the hint that when we increase LUT size the corresponding routing requirements do not grow in same proportion and hence the ratio of LUT (actual computation element of FPGA) to routing improves. Now if we analyze the last two columns (keeping these scenarios in perspective) some interesting observations can be made. The first column shows the area normalized to LUT4 architecture, the second column shows the observations of figure 4.3 for LUT mapping efficiencies. If we make a comparison now keeping LUT4 as a reference, an equivalent architecture based on LUT3 will be 0.8 times in area with logic mapping capacity of 0.75 this clearly shows LUT4 is a better choice compared to LUT3 as there is almost a similar result in terms of area however LUT4 will outperform LUT3 in terms of speed and power (since more logic in one LUT). Going now on the other direction, LUT6 architecture will be 1.7 times bigger on silicon and provides 1.54 times more logic capacity

compared to equivalent LUT4 architecture. Here we can observe a feeling of tradeoff between area and speed, switching to LUT6 from LUT4 will have a slight area penalty. But if we consider on more global perspective the LUT6 still seems more attractive as the area tradeoff is not so high and the price paid for area can have good tradeoff achieved by speed and power benefits of LUT6. Furthermore it also shows another example of FPGA CAD importance; an advanced mapping tool which can take more efficient use of LUT6 can outperform LUT4 in terms of area also. These are general observations; we will investigate a bit deeper in section 4.2. But from them also a general idea is obtained that why LUT4 still is a good enough general choice for overall nice tradeoffs of area, power, and performance among most vendors, and LUT6 (bigger LUT to trade area for speed and power, thanks to Moore's law) is common among biggest market share leaders in direct or adaptive form (Xilinx and Altera) as was discussed in chapter 2.

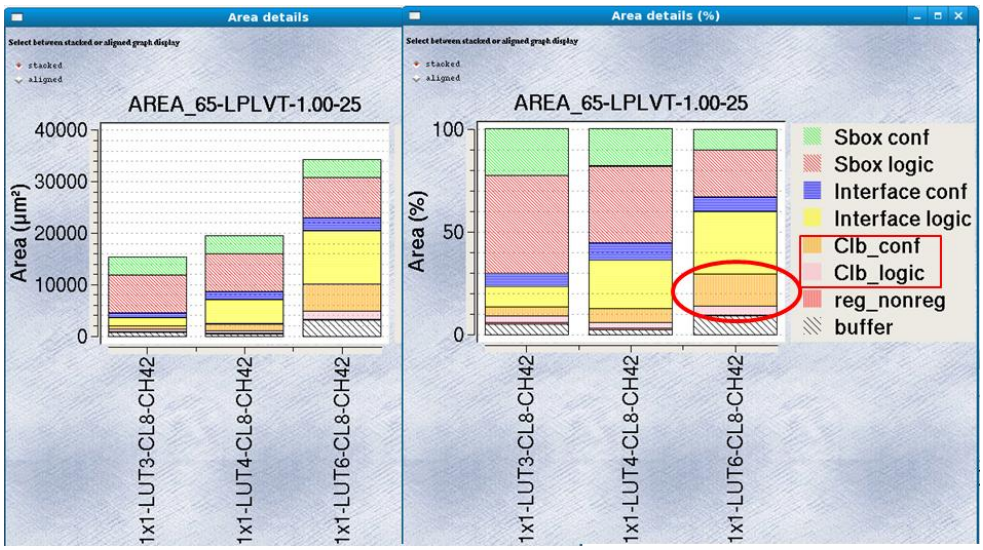


Fig. 4.4: LUT size comparison for area efficiency

LUT size	% of Conf.	% of Logic	% of Tile (Conf.+Logic)	Area Normalized to LUT4 arch.	Logic Capacity wrt LUT4 (mapping efficiency considered)
LUT-3	4.3	3.03	7.33	0.8	0.75
LUT-4	6.8	2.62	9.42	1	1
LUT-6	15.47	4.48	19.95	1.7	1.54

Table 4.4: Percentage of LUTs area in total Tile area and mapping considerations

4.1.3 Cluster and Channel

This section explains the exploration about the cluster and channel size. The LUT size exploration is closely related with the cluster size as different heterogeneous mix of combination and final interconnect architecture can highly influence the overall results. [Ahmed&Rose04, 4.8] explored that cluster size of 3 to 10 is a good value with LUT sizes of 4 to 6 (discrete homogeneous) and concluded that if an effective way of creating a cluster without getting the high area overhead of large LUT sizes (as we saw in last section) and the local multiplexing interconnect is developed, superior results can be achieved compared to using the classical architecture like they explored. Altera's ALM innovation [Altera_Lewis&Ahmed05, 1.22] has its motivations on such ideas. As this whole section (4.1) is discussing the basic explorations done in this work, this section describes the effect of channel size with cluster, the silicon exploration in that scenario and the effect of feedbacks. The experiments are done using LUT size of 6 and some with LUT size 4.

1- Channel width vs Cluster size

In a similar fashion like the LUT size exploration defined above, we implemented the 20 MCNC benchmarks for different cluster sizes (LUT size fixed at 4) and estimated the corresponding average channel width required for that cluster size (arithmetic average of minimum channel required for each benchmark for successful routing). Figure 4.5 illustrates the outcomes of the findings. We physically analyzed until cluster size 12 and extrapolated for some higher values based on the trends observed. We see from the graph that the channel size requirements do not grow in same proportion as the cluster size, e.g. going from cluster 4 to cluster 16 the channel width requirements approximately only double compared to 4 times increase in cluster size. This provides an interesting clue that the bigger clusters sizes can help exploit the benefit of this behavior as channel is often the most expensive part of the FPGA architecture. Next section describes the corresponding silicon explorations on ST65nmLP process and we will observe that in under experimented architecture the increased local multiplexing (SB-eLB muxes, figure 4.1) overrides the channel exploitation benefits of larger cluster sizes. Section 4.2 will address this issue (similar to Fc,in of scientific literature).

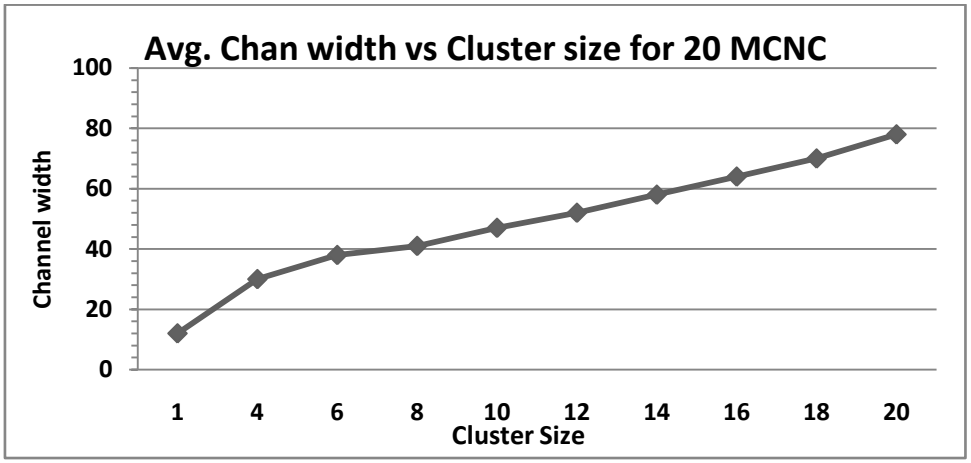


Fig. 4.5: Average Channel width (max) vs Cluster size for 20 MCNC

2- Silicon Exploration

Figure 4.6 describes some observations from silicon synthesis of different architectures with LUT size of 6 and different cluster sizes. The channel sizes were set closer or at suitable value based on CAD explorations. The graph on top left shows the area of the Tile along with contribution of different elements of tile, the graph on top right shows the percentage composition of different elements of Tile. The graph on the bottom shows the logic density (LUTs/mm^2) which is on one hand good measure of comparing architectures and on other hand essential element for eFPGAs.

Importance of Logic Density: For classical device based FPGAs logic density in many cases is not very critical or relevant for general description as they are characterized by the amount of logic (LUTs, hard blocks etc.) and I/Os that are available for the corresponding device. It is one of the reasons the logic density and die size etc. are mostly unknown and un-public for commercial FPGAs. However for eFPGA since it is an IP the comparison metrics and motivations are bit different and *logic density* of the architecture is a finer way of judging architecture for FPGA architects and SoC designers as it rapidly gives a rough idea about silicon feasibility and cross comparison of different architectures.

If we analyze the logic density exploration we observed that all the expected gains for higher cluster sizes were nullified by the sheer increase in local interconnect (SB-eLB muxes, mentioned as interface in figure 4.6) and buffering. The effect is more prominent as in the explorations in this phase there was no control of depopulation of SB-eLB multiplexers (as was discussed above). In section 4.2 we will investigate one step

deeper customizations enabled by eFPGA Creator and try to address this issue. But even at this raw experimentation level we can see that the eFPGA architecture is reaching around 300+ LUT6/mm² logic densities which is not very bad for a pure soft, technology independent architectures. [S-7][S-8] investigated eFPGAs in system perspectives using the best architecture found in these basic experiments (chapter 5 will address those experiments).

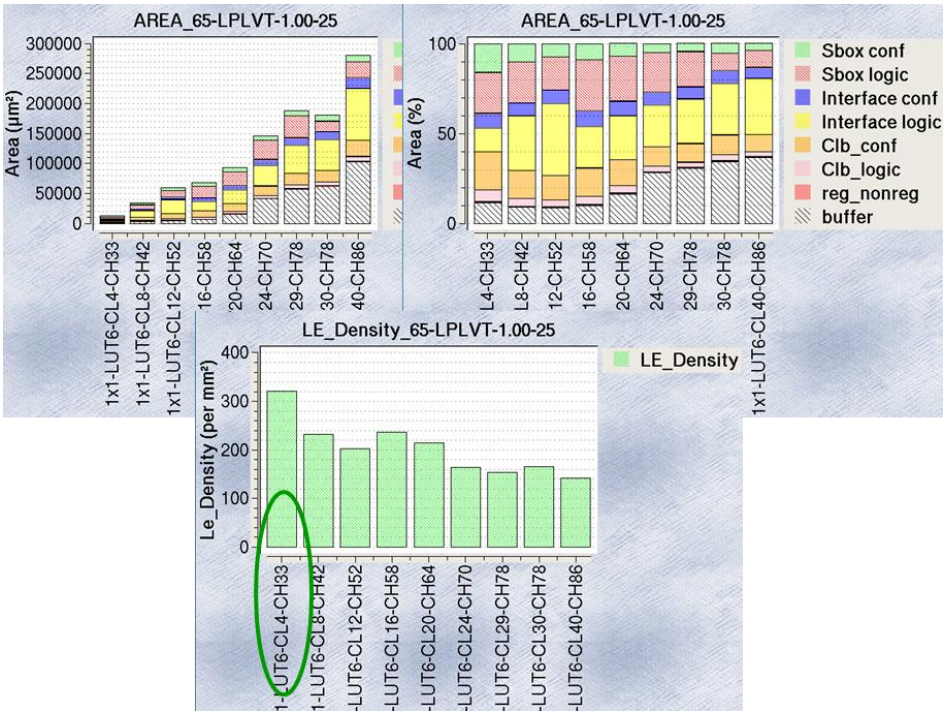


Fig. 4.6: Area comparisons for different cluster sizes

Figure 4.7 shows the results for power for these experiments (pessimistic power, section 4.1.6 will address the details about the method). The graph on the left hand side shows the plot of total power of the experimented tiles on 65nmLP process at 100MHz, 1.0Volts and 25 degrees temperature at default statistical toggle rates of Synopsys's Design Compiler (not accurate power but gives good approximate perspective for general overview and cross comparison). The static power is negligible in these experiments as the process is low power and temperature is 25 degrees (we will see these effects in later section on power). Indeed the power will go higher for a block as one increases cluster size, so to better visualize the behavior (similar theme as was done for area with logic density) the graph on the right side shows the power relative to cluster size 4 and we can observe a similar inverted view for power compared to logic density graph of figure 4.6 (more gates means more power) and see that the cluster 4 is relatively providing better power efficiency.

Figure 4.8 shows the timing statistics of the experimented tiles (Note: the critical path of tile by Design Compiler, it is not the application critical path; in section 4.2 we will explore real application delay). The behavior is self explanatory, we can observe that as we increase the cluster size the amount of logic in the critical path in tile increases because the size of the SB-eLB multiplexers increase with the increase of cluster size. A relatively strange behavior was observed for cluster 29 and 30 (can be treated as experiment noise and neglected) where instead of getting higher compared to cluster 24 it actually dropped. We tried to investigate the behavior and what was found as a reason was that Design Compiler chose different kind of multiplexers and buffering for these cases which changed the overall behavior. As to why it did that we got no plausible explanation, however this gave an idea that much can be played around with the silicon tools options and constraints to achieve different behaviors for the same hardware.

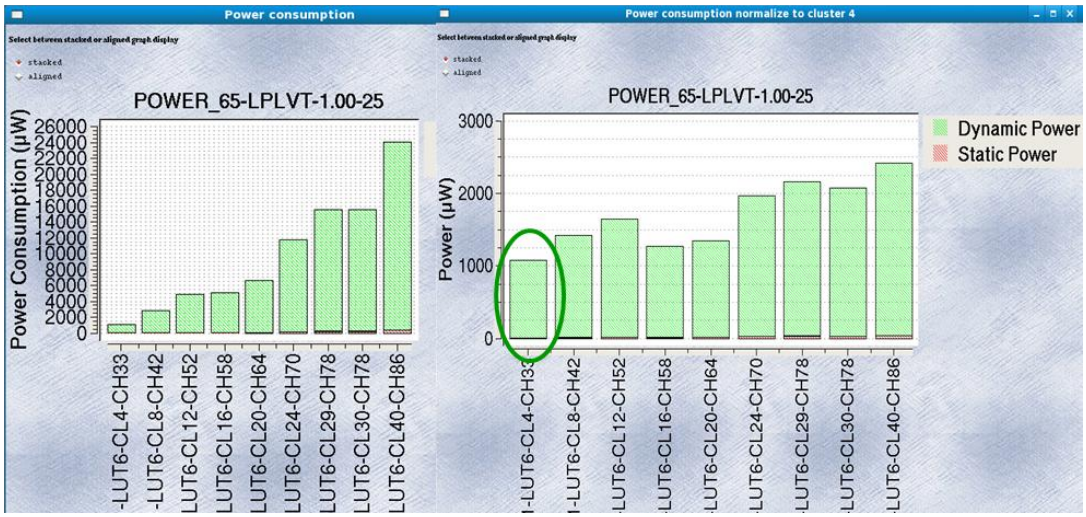


Fig. 4.7: Power comparisons for different cluster sizes and relative to cluster size 4

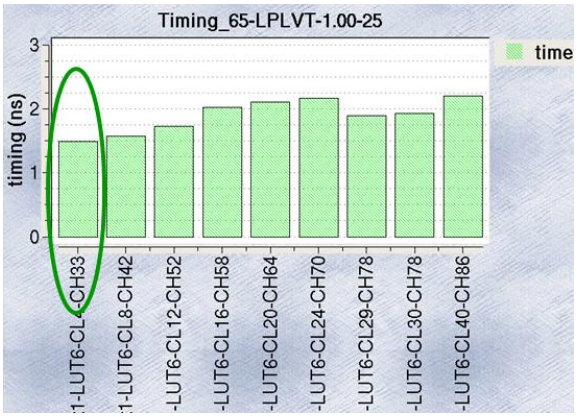


Fig. 4.8: Timing comparisons for different cluster sizes

3- Clustering Feedback analysis

In the previous section we analyzed different cluster sizes and in silicon exploration observed that the interconnect multiplexers (SB-eLB) to provide a unified connection block plus fully connected local interconnect (logically similar to figure 2.21 in chapter 2) have a large silicon overhead. There are two contributors to the inputs of those multiplexers (figure 4.1). First the larger part is the inputs to the cluster from outside world (routing tracks). Second relatively smaller but yet area consuming part is the feedbacks of the cluster. To provide a fully connected structure each output of BLE (LUT+FF pair) is fed back to every single input of every BLE (figure 4.1). This section describes the feedback statistics to get a feeling of over flexibility of fully connected feedbacks.

Figure 4.9 shows the feedbacks statistics obtained for different cluster sizes (clustered with TVPack [4.1]) using Clustering Analyzer tool. The results will vary based on clustering CAD algorithms but the general trend might be similar. We observe from the graph that there is a steady increase in number of feedbacks as the cluster size grows. However what is significant to note is that the amount of feedback compared to hardware flexibility provided by a fully connected structure is very low. For instance if we take the case of cluster size 10 as example, the graphs shows that on average for all benchmarks the amount of feedback in the cluster is around 6, which is fed to 40 inputs (10x4, Lut size in these experiments is 4) of the 10 LUT4 in the cluster where 34 (40-6) are a waste of hardware flexibility. The graph also illustrates average statistics of distribution of the feedbacks in the cluster, for the analyzed case the 6 feedbacks in the cluster are used by 4 distinct LUTs meaning a good split distribution with some LUTs having more than 1 input which is feedback from the cluster outputs.

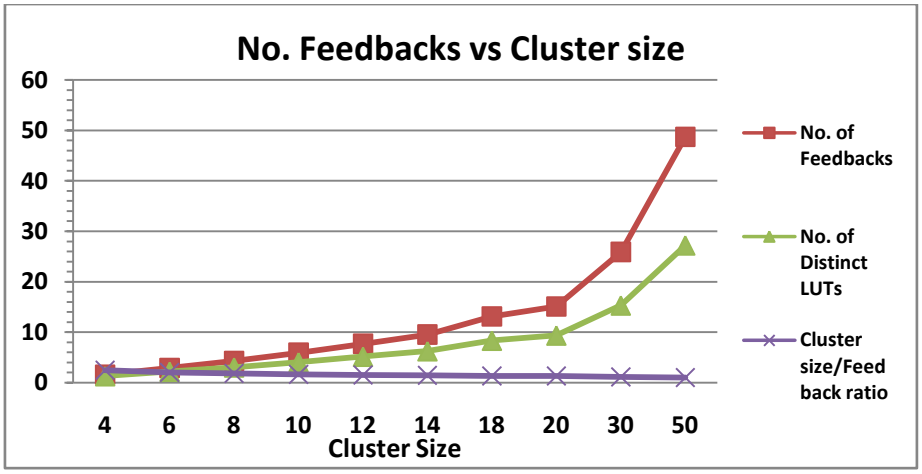


Fig. 4.9: Average feedback statistics for different cluster sizes for 20 MCNC benchmarks

4.1.4 Routing Analysis

Routing in general is the most expensive and in many ways most challenging and exciting part of FPGA research. This section briefly describes the explorations done on different aspects of routing analysis. It first describes the challenges of fluctuating channel size demands for different applications and their corresponding challenges for FPGA design decision to cope with that. Second an overview of the channel utilizations efficiency, challenges for mapped applications is discussed. Third it provides an overview of data traffic in tile for architectural enhancement clues and finally hop analysis to investigate the benefits of heterogeneous routing with long and diagonal wires.

1- Channel size challenges

The channel size requirements of the FPGA highly vary from application to application. While conducting architectural exploration, often one of main objective is to find the minimum routing channel needed for target application. [Betz&Rose99, 4.1] proposed the method of binary search to find the minimum routing requirement for the mapped application and use 20% more routing than minimum requirement as a tolerance. Finding minimum routing requirement for each application is essential for FPGA architects as it helps in several decisions, this style has remained similar in several VPR based/like research works from many researchers. However it does not fully represent a real life FPGA challenges scenario, where an FPGA has to be capable of dealing wider varieties of applications. [Altera_Lewis05, 1.22] describes the challenges with the state of art FPGAs regarding channels. As the commercial FPGA needs to be able to route all the commercial benchmarks and not the average (like often found in academic literature) so a higher weight is put on the largest channel hungry benchmarks. Because a single benchmark of very high requirements can make the architecture too expensive they do not strictly use absolute maximum but allow some discretion. To compare architectures a metric of satisfying around 99% of benchmarks is used. This illustrates how challenging this issue is in real-life scenarios.

Figure 4.10 shows the minimum channel size requirements of the 20 MCNC benchmarks for different LUT4 based clusters of different sizes (1, 4, 6, 8, 10, 12); we can observe how fluctuating the requirements are for channels for different benchmarks. Furthermore if we have a look at the statistics of the benchmarks from table 4.1 and analyze the graph of figure 4.10 we can see that there is no easy or apparent metric of judging the channel requirements of a benchmark based just on its size or I/O requirements. Regarding the discussions concerning state of art above, if we have to create FPGA architecture able to target all these benchmarks e.g for cluster size 12 (c12) we need channel size of around 80 (required for ex1010 and clma) to pass all benchmarks. A choice of channel around 50-60 will satisfy majority of them (even in that case many are over served in terms of channels). [Tom&Lemieux06, 4.18] proposed an interesting way using CAD to implement un-routable designs due to channel restriction at the expense of using/wasting some extra LUTs by depopulating the clusters of congested areas. Furthermore if we compare the results of

figure 10 with other research works for uni-directional routing (e.g. [Luu&Rose09, 4.5]) we observe that the channel demands are much similar for our architecture although being fundamentally different on some aspects (no CB and no logical equivalence exploitation of eLB inputs and outputs due to eliminated classical local interconnect).

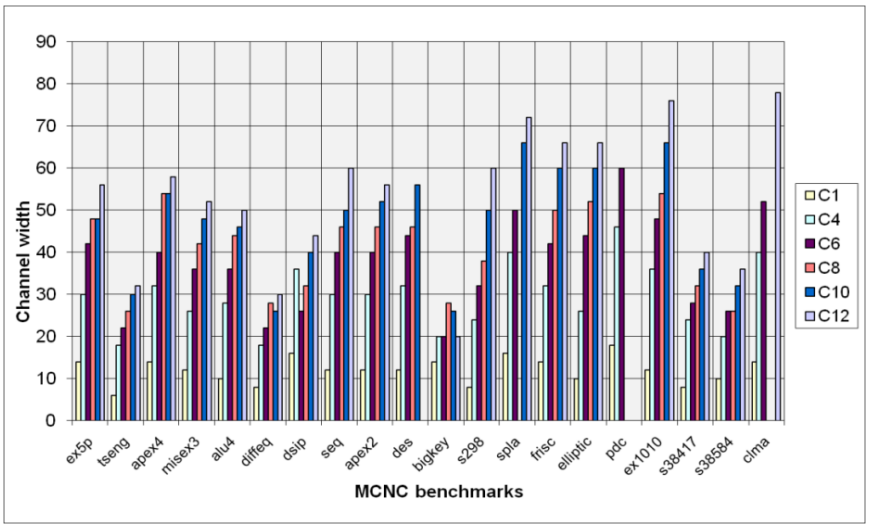


Fig. 4.10: Min. channel width requirement for different cluster sizes for benchmarks

2- Routing efficiency analysis

In previous section we discussed the channel variation challenges across benchmark applications. This section describes a somewhat related issue about the routing statistics of mapped applications which also reveal some interesting observations that are not well addressed in academic literature. In last section we observed that the benchmark having the highest routing demands set the mark for the worst case scenario and routing architecture is designed to cope with it. In a similar fashion when an application is mapped on an FPGA, the utilization of channels is highly heterogeneous due to obvious reasons as unlike hardwired ASIC design where router *creates* wires for routing, the FPGA has pool of routing resources and router has to *use* these resources. The highest demanding channel requirements (highest congestion area/areas) create the bar of the channel size requirements for that benchmark application. Figure 4.11 shows a snapshot of channel efficiency analyzer tool which analyzes every benchmark and overall statistics of benchmarks (figure 3.21 in chapter 3 is enhancement of this tool). We can see that on average (keeping in mind in addition that all benchmarks are analyzed for architecture with the minimum channel required for them!) there are large amount of channels which are partially packed, only a small percentage of high requirement channels (in worst case can be a single channel/track) create the requirement threshold and whole architecture since FPGA has to be generic (or as generic as possible) is created with that channel size.

3- Tile traffic analysis

The tile traffic analysis explores the data traffic flow in the tile. It investigates how the input traffic of eLB is distributed in the channels of SB in four directions/ports (North, South, East, West as shown in figure 4.1) and how the outputs of eLB are distributed in the channels of SB. These investigations gave basic clues about the optimization areas like the SB-R and SB-eLB multiplexers (similar to Fc,in and Fc,out). Figure 4.12 shows a snapshot of the traffic analysis tool of Routing Analyzer which provides average statistics of different benchmarks and overall statistics of a set of benchmarks (figure 3.20 in chapter 3 is enhancement of this tool). Section 4.2 will address in detail the architectural enhancements and explorations in this regard. The high waste of routing resources (normal and inevitable in FPGAs) for a chosen application can be visualized in the figure (red boxes) highlighting several routing tracks in connection with eLB inputs and outputs are never used (motivations for routing architecture depopulation for increasing silicon efficiency).

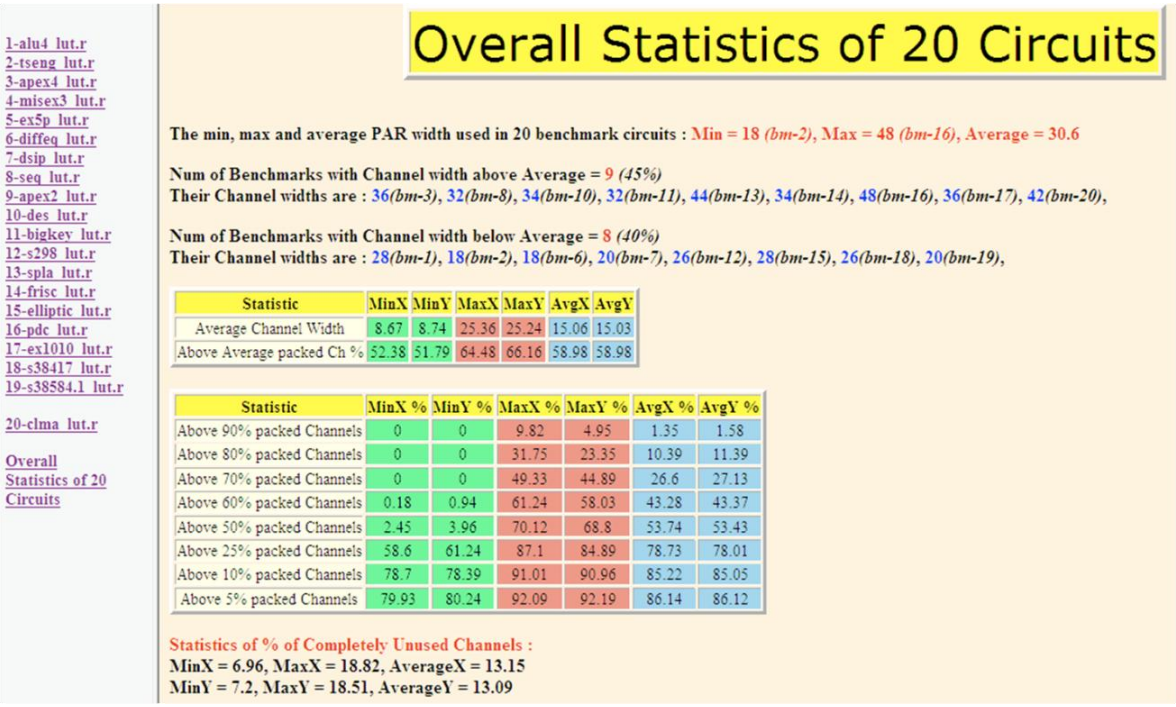


Fig. 4.11: Routing efficiency analysis statistics

Fig. 4.12: Tile traffic analysis

4- Hop Analysis

Hop analysis helps to get clues about the heterogeneous long and diagonal routing wires. Extending the concept of tile traffic analysis which investigates intra-tile traffic, the hop analysis analyzes the inter-tile traffic patterns to visualize the nature of traffic in the architecture for benchmark applications. Figure 4.13 shows a basic diagram to illustrate the hop for single length (length 1) architecture by showing the Manhattan distance a signal can travel on routing architecture in a single hop (jump). The figure is intuitive, in a single jump the reference tile (shown as 0) is connected to direct four neighbors (shown as 1), in two hops the data (crossing trough the first hop tiles) can reach to eight tiles (shown as 2) and so on. In case of heterogeneous long and diagonal wires the data can directly (long jump) reach to other tiles, enhancing the performance. The objective and challenge of hop analysis is to explore how much benefit

can be achieved with long hops and how much should be the ratio/quantity of heterogeneous wires to make an area efficient solution while providing enhanced performance benefits of longer hops. Figure 4.14 shows the detailed statistics observed for 20 MCNC benchmarks (figure 3.22 in chapter 3 is enhancement of this tool). The pattern observations are summarized in the table; we can see how much traffic was on average at different hop levels. e.g. if a length-2 long wires and length-1 diagonal wires are used they will enhance the data reach to almost double (25%) compared to only single hop length-1 architecture.

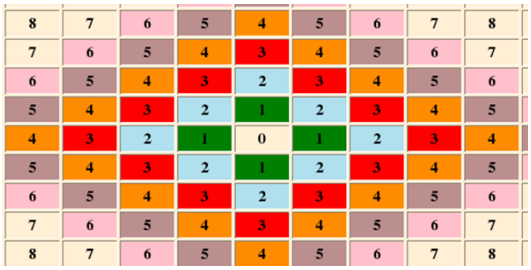


Fig. 4.13: Routing Hop

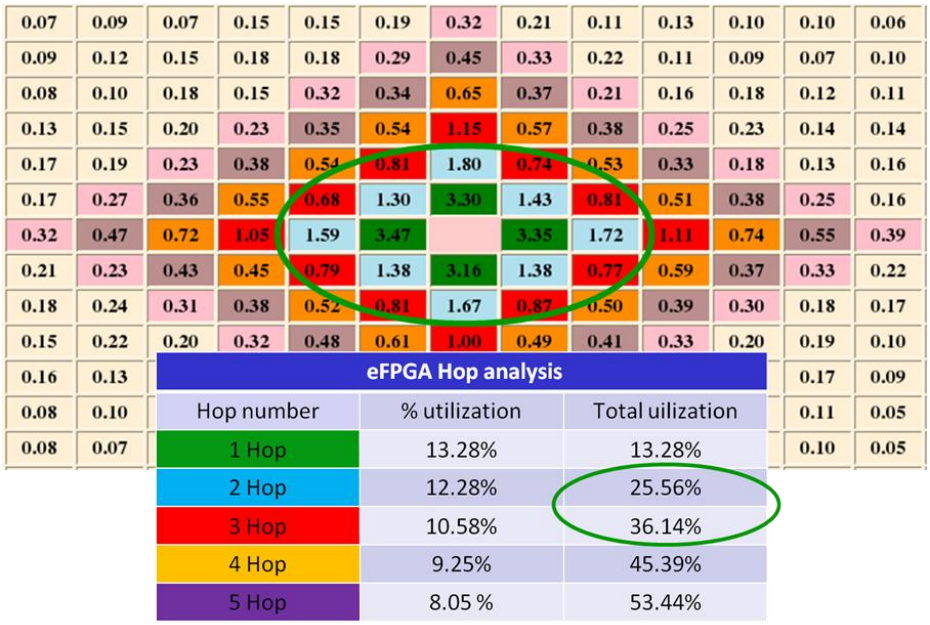


Fig. 4.14: Hop Analysis

4.1.5 Challenges of Power Consumption

This section describes the general explorations to investigate the effect of process technology scaling in beyond 130nm nodes (where voltage scaling is flattening, static power becoming major concern) and the effect of threshold voltages in power vs speed scenarios.

1- Area, Power, Speed comparison at 90, 65, 45nm nodes

At these finer nodes there are a wide range of library options provided by Fabs e.g. GP (General Purpose), LP (Low Power), several threshold voltages (low, standard, high) etc. to cover a wide range of consumer requirements based on target markets. For our experiments we had an access to only a subset of libraries of ST provided through CMP. The following libraries were used: 90nmGP, 65nmLP and 45nmLS. Figure 4.15 shows the silicon statistics of a tile (LUT-4, cluster-4, and channel-42) on different nodes. Some interesting observations can be made from graphs. Firstly for area we observe the Moor's law smooth effect of doubling the transistors. The gain from 90nm to 65nm is slightly less than 2X in the graph. We investigated the reason and found that in the 90nm standard cell library there was Mux 8x1 while in 65nm and 45nm library Mux 4x1 is the largest Mux. As the eFPGA has highly Mux dominated architecture the lack of larger Mux in the libraries turned out to be a slight disadvantage from silicon perspective. Now if

we observe the graphs for Power and Timing things are much different compared to nice improvements in Area. We can observe that supply voltage is not scaling down like past in finer nodes hence limiting the power scaling benefits. It is important to consider that the library for 90nm is GP while those of 65nm and 45nm are LP, GP process is faster compared to LP but has high leakage as a tradeoff, hence higher power consumption.

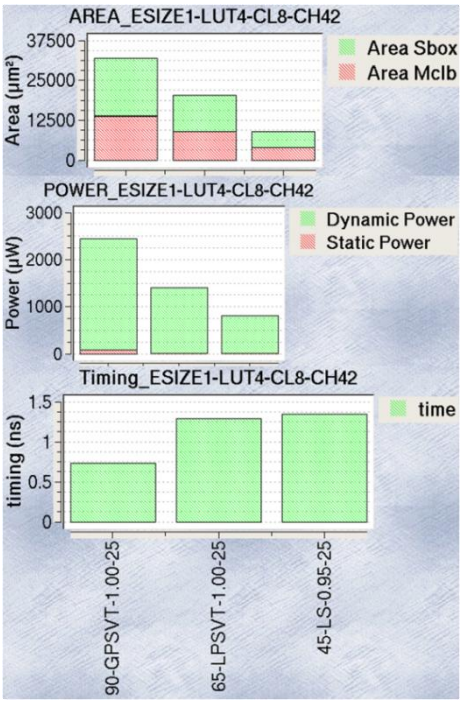


Fig. 4.15: Comparative Area, Power and Timing comparison on 90, 65 and 45nm

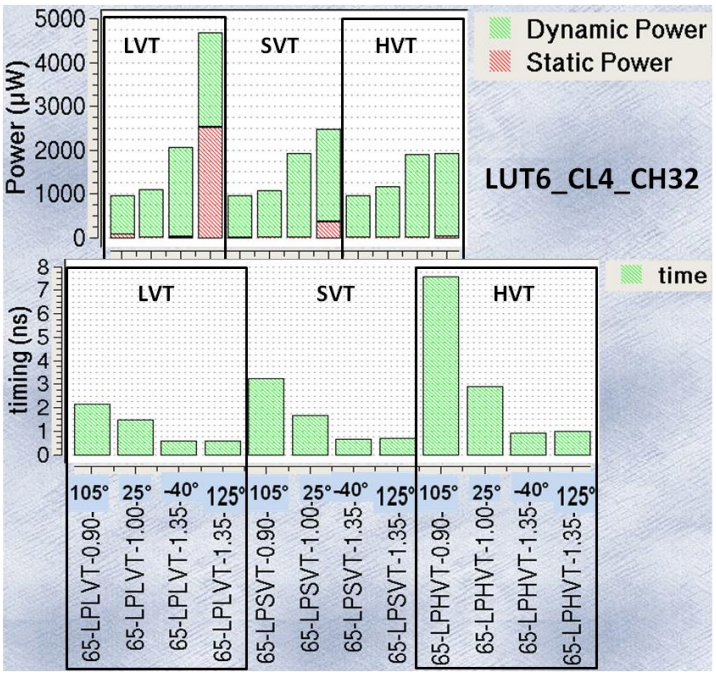


Fig. 4.16: Power vs Speed with different threshold voltages

2- Power vs Speed: effect of threshold voltage

In the previous section we tried to observe the effect of process scaling on area, power and speed. This section shows highlights of the effect of threshold voltage on power and speed for a fixed node. We conducted experiments on 65nmLP in this regard. Figure 4.16 illustrates the explorations. The results are

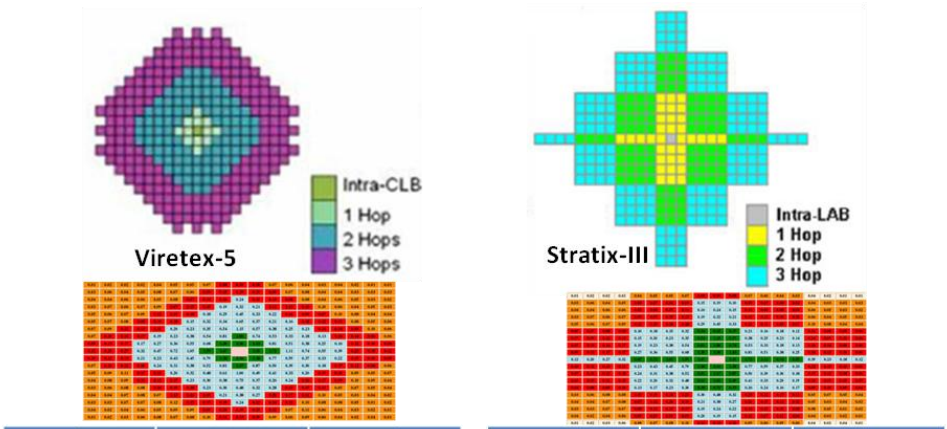
divided in three categories based on low, standard and high threshold (LVT, SVT, HVT) respectively. For each category different experiments are illustrated showing effect of supply voltage (0.9, 1.0 and 1.3 volts) and temperature (-40, 25, 105 and 125 degrees) variation using the best, nominal and worst case libraries. As the process is LP, the static power is not very high, however we can see the effect of voltage and high temperature which shoots up static power as it has strong dependence on them as was discussed in chapter 2.1.2. The interesting part to observe in these graphs is the variation of power and speed with the change of threshold voltage (power vs speed). Threshold voltage significantly changes the static power (leakage). For a distinct comparison we can observe that strong shoot of static power at a worst case (1.3V, 125degrees) is almost eliminated by increased threshold voltage however the benefits of higher threshold for power come at a price of speed which is significantly higher for high leakage transistors. That highlights another aspect of challenges of power. In chapter 2.1.2 we observed how state of art deal with these issues with a complex mix of transistors and threshold voltages to achieve better tradeoffs in this regard.

4.1.6 Comparison with State of Art

In the last phase of the basic explorations we used some of our results and tools to compare our findings to state of art FPGAs to find motivations and get a rough idea of how we compare to state of art. Below two major explorations done are briefly described.

The HOP

Figure 4.17 shows the experiments of HOP analysis for patterns of Xilinx's Virtex-5 and Altera's Stratix-III which were observed in chapter 2 (figure 2.13). Although the experiments cannot be fully correlated with Xilinx and Altera devices and judgment of one over another but provides good basic understanding of potentials and comparison of different hop styles. The analyses are done using our Hop Analyzer tool; we added some features in the tool to display the hop results like the patterns of Xilinx Virtex-5 and Altera Stratix III. It is important to note that there are no physical long wires in the explored architecture; the architecture is purely homogeneous with only fundamental single length wires. The benefit of analyzing like that is that we get an interesting rough idea of potentials of different hops, which helps to move to next step of physically deciding the length and quantity of long wires.



HOP Number	% Utilization	Total Utilization
1 Hop	25.56%	25.56%
2 Hop	40.27%	65.83%
3 Hop	13.94%	79.8%

HOP Number	% Utilization	Total Utilization
1 Hop	40.65%	40.65%
2 Hop	28.23%	68.88%
3 Hop	14.62%	83.5%

Fig. 4.17: Hop Comparison of Virtex-5 and Stratix-III for inspiration

In figure 4.17 on top we can see the hop patterns (physical) of Virtex-5 and Stratix-III as were discussed in chapter 2. Analyses of the average statistics of 20 MCNC benchmarks are shown in the tables. We can see that for single hop Stratix-III has higher data reach compared to Virtex-5, going higher both devices have similar characteristics. This also provides the inspiration of the diagonal long wires innovation of Virtex-5 in addition to classical long wires which are common in all commercial FPGAs.

Core Power Consumption

Finally we made some comparisons of our results about power consumption (both static & dynamic). Power while being the hardest challenge for research and industry at present is also the hardest element to evaluate and analyze compared to area and speed. However getting some basic idea was essential to have a feeling. The soft technology independent nature of eFPGA helped in this regard. [Altera_Video, 1.25] describes a comparison of core power consumption of Altera's EP3SL150 and Xilinx's XC5VLX110 which are 65nm devices. This provided a help to get an idea of power consumption in state of art, furthermore as it demonstrates the core power this provided additional help as pads (I/Os) of device stand alone can consume significant power and are not main concern of this work as eFPGA is an IP. The power consumption of the two devices is computed physically (core supply voltage and current) by mapping several benchmarks from opencores (opencores.org) to fill the devices. The reported total core powers of the devices are reported as **1.9W for EP3SL150** and **2.3W for XC5VLX110** respectively at **100MHz**. We took the Altera results for next step as the demonstration was from Altera. We used the EPE (Early Power Estimator) tool utility of Altera which is freely available on their web to find the core static power of EP3SL150 which was found to be **269mW** at 25 degrees.

Pessimistic dynamic power: As dynamic power highly depends on the transition activities, it was very hard to get equivalent results. For that we made a pessimistic approach taking benefit of soft technology independence nature of eFPGA to get a rough idea for inspiration. Firstly in order to be fair in comparison we estimated the power consumption of an eFPGA with 150,000 LUT-6 (cluster 4, channel 32) which is lot more than 57000 ALMs of the target device, we did that because Altera device has several hard macro blocks (memories, multipliers etc.), routing architecture is much deeper, it will not be fair if we just compare on number of LUT bases. Secondly to be further pessimistic in order to get worst case figures we estimated power at different toggle rates (TR) and static probabilities (Stp) for a single Tile and multiplied the result by total tiles. Table 4.5 presents the results that we obtained for static power (SP) and dynamic power (DP) at 65nm (same target node). We can see that our initial results of eFPGA are not so bad. Because there is no kind of power management in eFPGA at the moment and it's a purely soft core with raw customized architecture compared to highly optimized full custom Altera FPGA and also not all tiles in eFPGA will be switching at the same level. Furthermore eFPGA is supposed to be very small (few thousand LUTs at maximum). This provided motivations for future innovations and foundation of system integration evaluation experiments of [S-7][S-8] (Chapter 5 will address that).

65nm LP process. 1.0V, 25 degrees. 100MHz	LVT	SVT	HVT
Static Power (mW)	382	31.5	3.3
Normal: Dyn. Power (W) @ (Tr-0.25,Stp-0.25)	6.9	6.732	7.794
High: Dyn. Power (W) @ (Tr-0.50,Stp-0.5)	14.29	13.99	16.14
Very High: Dyn. Power (W) @ (Tr-1.0,Stp-0.50)	28.61	28.01	32.33

Table 4.5: Pessimistic Power of 150000 LUT6 eFPGA at different threshold voltages

4.2 Explorations with Tile Customization (eFPGA Creator)

This section provides second level of research explorations of this thesis work for eFPGA architectural exploration. In section 4.1 we covered the basic fundamentals which in addition to survey knowledge of chapter 2 provided practical knowledge and motivations for the research work. As was discussed in introduction of section 4.1, in basic explorations we were restricted to only higher/coarse grain customization experiments due to limitations of CAD tools and adaptable VHDL. Compared to previous section where SB was almost a black box in a way we could not change anything other than channel size, this section specifically focuses on SB customization (for a comparison with scientific literature in some ways similar to the Fc,in and Fc,out research challenges). Section 4.2.1 provides general motivations along with some further architectural challenges which are more appropriate and easy to discuss at this point. It provides the experimentation methodology, tools, benchmarks used for experiments. Sections 4.2.2 and 4.2.3 are the core of this section and describes detailed experiments for SB-R and SB-eLB multiplexers (figure 4.1) respectively, section 4.2.4 briefly address the combination of best results of these two sections to find the optimal architectures of this research work. Finally section 4.2.5 addresses some further investigation frontiers which will also provide ease for some of the discussions of chapter 6.

4.2.1 Experimentation motivations and methodology

This section provides fundamental motivations for these second levels of experiments and the experimentation methodology for the experiments by explaining the CAD tools and benchmarks.

1- General motivations

Since the motivation behind this thesis work is to investigate soft eFPGAs, this in some aspects make/can make the exploration challenges and issues relatively different than full custom layout based FPGAs which are common in all commercial devices and academic research (VPR/VPR-like) have also mostly focused on full custom device based FPGAs as they are mainstream. In case of soft eFPGA all the switching is done through standard cell library multiplexers of target node (no pass-transistors/tri-state buffers etc.). In some manners it can change the architectural scenario significantly.

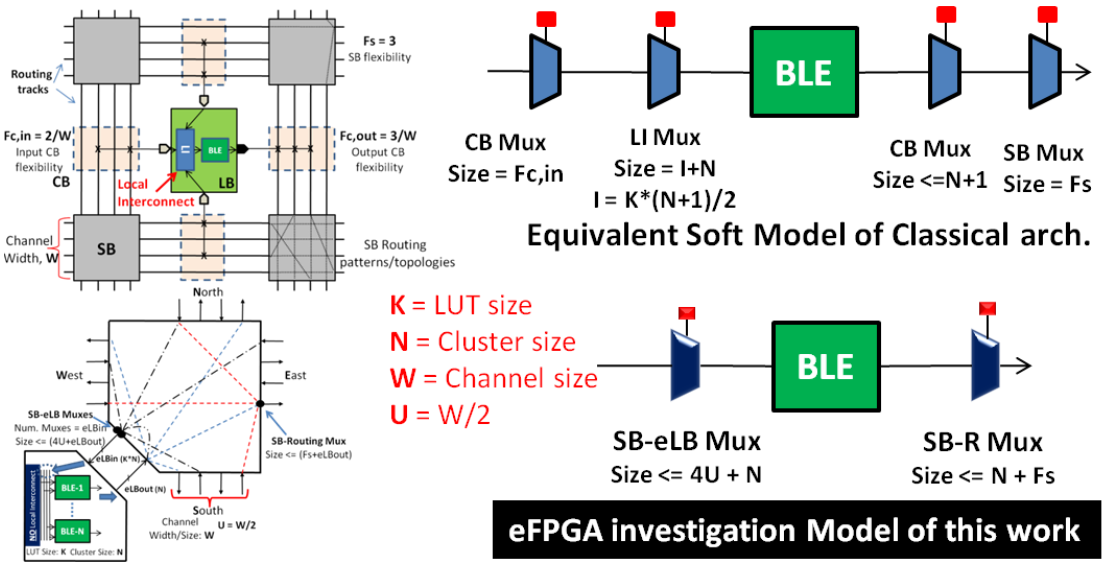


Fig. 4.18: The equivalent comparison of classical soft and soft eFPGA architecture

Custom vs Soft Hardware: Figure 4.18 theoretically explains the scenario of a classical CB based architecture equivalent (clone) soft eFPGA architecture model and the unified SB based soft eFPGA model that is investigated in this thesis work. If we briefly analyze the figure some interesting observations can be made which also highlight the pros and cons of the two styles. For making a rough equivalent comparison

assume a LUT size of 4, cluster size of 4 and channel size of 48 (are quite realistic parameters). We assume Fc values to be 1 (for input and output) and Fs = 3. If we analyze the two models of figure 4.18 we come across following results.

- **Classical:** Muxs, CBin = 48bit, LI = 14bit, CBout <= 5bit, SB = 3bit; Conf = (6+4+3+2)bits
- **Unified SB:** Mux, SB-eLB <= 100bit, SB-R <= 7bit; Conf = (7+3)bits

These are the worst case possible results for the two cases. If we take a closer look at the results and figure 4.18 we can see that for the worst case the unified SB model is larger in terms of area (immense sized SB-eLB multiplexer) but will be relatively faster than classical approach as the total mux depth (10bits) is lesser than the classical architecture (15bits). Another big difference between the layouts based architecture and soft counterpart is that in custom layout the CB multiplexers are implemented (often) with pass-transistors. This while logically creates same scenario, is practically quite different compared to multiplexers. In custom layout the CB size (Fc,in and Fc,out) do not have a significant effect on speed due to obvious reason of pass-transistors based switching [Betz&Rose99, 4.1][Marquardt.Betz.Rose00, 4.10]. However in case of multiplexers the delay will significantly vary as CB muxes depth will vary with Fc value. This provides some initial motivations to explore the potentials and customizations of unified SB model if it can have better silicon efficiency compared to classical model (in soft eFPGA scenario).

Inspiration from State of Art: The classical CB based custom architecture shown in figure 4.18 is based on VPR architecture model which is/has been widely used and experimented in research community and most of the works have tried to improve some aspects of it without/rarely making any fundamental change. If we analyze the state of art several different kinds of topologies and styles exist particular to the vendors. VPR is closely related to Altera (due to historical reasons [1.24]). In [1.22][1.26] Lewis et al have explained Altera architecture based on enhancement of VPR to state of art. Appendix A1 provides examples of state of art FPGA patents, which have some aspects similar but very different architecture styles. They also provided some additional motivation to explore some different what-if cases.

Unified SB inspirations: Some of the motivations described above and specific exploration flexibility and ease enabled by eFPGA creator provided an inspiration to experiment some different architectural scenario compared to re-addressing and improving the classical VPR model architecture. The unified SB architecture is partly similar to classical architecture that is widely addressed in research. These relatively new kind of experiments are less addressed in research due to somehow VPR wide sense limitations and will be a good contribution to community for observing and cross analyzing some new what-if scenarios. The next section describes some of the key exploration objectives and motivations behind them.

2- Exploration objectives

The key objective of these explorations is to customize the SB-R and SB-eLB multiplexers as shown in figure 4.19 to create more silicon efficient architectures and observe how competitive this solution is compared to CB +LI based classical architecture. The major objectives can be described as follows.

- How suitable/penalizing it is to completely remove the fully connected local interconnect (LI)
- The effect of SB-R and SB-eLB multiplexers customization on silicon efficiency
- The effect of SB-R and SB-eLB multiplexers customization on routing flexibility (channel width requirements for successful routing)
- The effect of SB-R and SB-eLB multiplexers customization on routing efforts (routing time, routing iterations change due to decreased flexibility of routing because of customization)
- What are the best ways/styles to create efficient custom architectures
- The algorithmic friendly ways of efficient customizations to augment them easily into eFPGA Creator for wider design space exploration
- Exploration in discrete manner to clearly identify effect of each customization

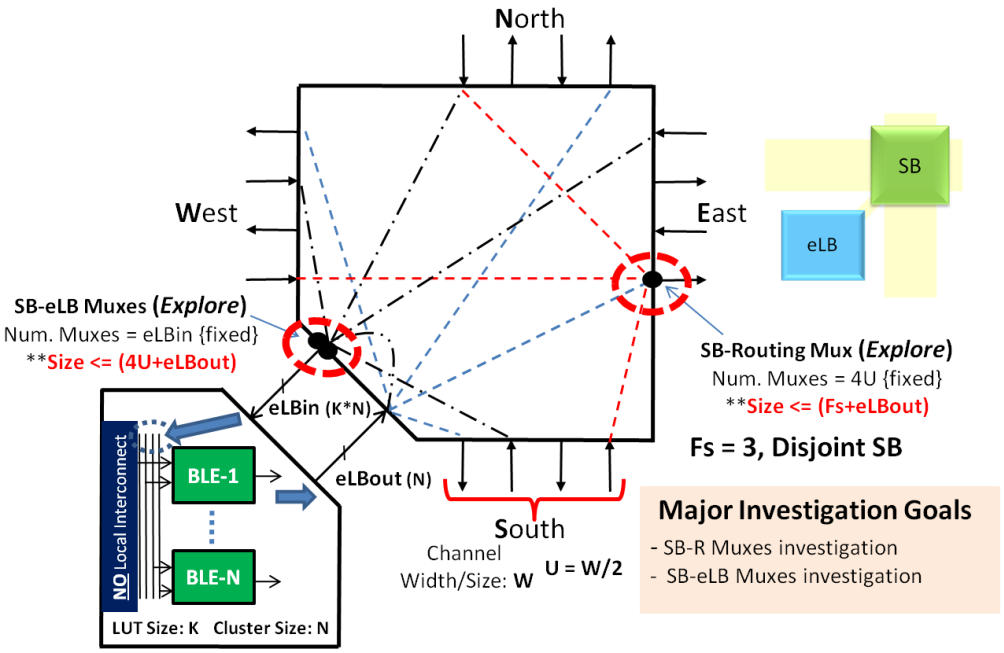


Fig. 4.19: The investigation objectives of SB customization

3- CAD flow and Benchmarks

These experiments were conducted using the eFPGA Creator and eFPGA Programmer tools that we discussed in chapter 3 (the tools from 3Q 2010 version were used for these experiments). The customized tiles (SB, eLB) were created using the graphical infrastructure of library manager (chapter 3). The hardware of eFPGA was auto generated and silicon implemented by hardware generators of eFPGA Creator (chapter 3). The target node used for the experiments was ST65nm LP-SVT process. The timing information about the experimented architectures for eFPGA Programmer are obtained by the extractions of timing (.sdf, .v files of Design Compiler, only front-end was used in presented experiments) information from silicon flow. The experiments were focused on a LUT size of 4, cluster size 4, channel size 48. The eFPGA cores of 1024LUTs and 256 I/Os were created for every custom tile for experimentation using the architecture manager (chapter 3).

For the benchmark applications we selected 10 benchmarks from section 4.1 and added one more MCNC benchmark (k2). The 11 benchmarks that are used in the following experiments are shown in table 4.6. We can notice a decrease in the number of LUT4 for the benchmarks compared to table 4.1 (SIS mapping). The enhanced quality is due to the mapping done by eFPGA Programmer which uses a slightly modified version of ABC mapping tools of Berkeley University [4.83][Brayton&Mishchenko10, 4.86].

S No.	Circuit	inputs	outputs	I/Os	LUT4
1	**ex5p**	8	63	71	240
2	**tseng**	52	122	174	743
3	**apex4**	9	19	28	742
4	**misex3**	14	14	28	426
5	**alu4**	14	8	22	614
6	**diffeq**	64	39	31	714
7	**seq**	41	35	76	781
8	**apex2**	38	3	41	714
9	**k2**	45	45	90	546
10	**s298**	4	6	10	856
11	**ex1010**	10	10	20	922

Table 4.6: Selected benchmarks for tile customization experiments, mapped with eFPGA Programmer

4.2.2 SB-Routing Multiplexers optimization

This section details the explorations of different customization ways of SB-R (routing) multiplexers optimization. It first describes the experimented custom architectures properties and then the silicon implementation and benchmark mappings results are presented in detail.

1- Custom Architecture types

The SB-R multiplexers (figure 4.19) are the routing multiplexers which allow switching of different routing tracks inside the SB and transmission of outputs of eLBs to the routing network. The objective is to reduce the size of these multiplexers in such a way that more silicon efficient (area, power, speed) architecture is created with minimal tradeoff to the routing efficiency for mapped applications. For experimentation seven distinct types of customizations were investigated based on the knowledge of chapter 2 and section 4.1. All of them have LUT size 4, cluster size 4. Choosing a fixed LUT size and Cluster size for exploration is also a what-if motivation for specifically investigating a scenario where some fundamental parameters are fixed and for them best routing architecture is explored. The inspiration partially came from state of art where often the fundamental parameters are fixed to an apparent magic value which are often similar (LUT4, Cluster4; LUT4, Cluster8 etc. which do not change often/abruptly in different generations of architecture) while the routing architecture varies.

They can be divided in three themes which are discussed below and shown in figure 4.20 in detail with their architectural properties (including algorithmic description for more complex structures).

Base (Full)

tsr_F: Represents the base for reference with highest flexibility for routing. This case was used throughout section 4.1 in the presented experiments and represents the worst case in section 4.2 experiments.

Theme/Strategy 1 (cutting ports partly/fully)

The customizations of this theme are partly inspired by our state of art discussions in chapter 2 (figure 2.10) for routing architecture sides. It addresses the effects of cutting eLB outputs partly/fully from ports.

tsr_noN: The traffic of eLB outputs is fully cut from North (N) port, {three-sided routing}.

tsr_noNE: The traffic of eLB outputs is fully cut from North (N) and East (E) ports, {two-sided routing}.

tsr_50pAll: The traffic of eLB outputs is distributed to 50% of all four ports. In principle it is equal to tsr_noNE scenario, instead of fully cutting two ports 50% of each port is used.

Theme/Strategy 2 (diverse customization)

The customizations of this theme address diverse customization in different forms in such a way that every SB-R multiplexer has one eLB output connected instead of all. As the Fs value is fixed to 3 this provides a near perfect scenario from multiplexer size point of view making it an optimal value of 4 from configuration (full 2 bit) stand point. This is also an interesting motivation issues (particularly for soft eFPGA as its harder to make custom fast paths/mux inputs which are common in state of art) which is/has been often ignored in research works, adding a single more input (increase Fs or eLBout) will lead to abrupt move to higher underutilized configuration value (3 bit) and delay penalty of increased multiplexer depth. Since SB-R multiplexers are huge in terms of quantity (channel size are usually high for realistic architectures) the penalty of area and delay (critical paths often pass through several SB) is multiplied.

tsr_c1: The eLB outputs are distributed as one on each port. This represents the similar classical scenario in VPR architecture where each LB pin is only connected to one side of routing tracks through input or output CB (relatively more layout friendly way for full custom).

tsr_c2: The eLB outputs are smoothly connected to all four ports in the manner shown mathematically in figure 4.20. This highlights a differentiating factor of eFPGA architecture with unified SB enabled by eFPGA Creator and eFPGA Programmer to exploit. In conventional VPR architecture it is not possible to create such connections as each pin of LB is dedicated to one side through a CB prohibiting such type.

tsr_c3: In a similar motivation like tsr_c2 the eLB outputs are distributed in further diverse and interleaved manner explained mathematically in figure 4.20 along with some issues related to channel size

For all the cases in the seven custom tiles the SB-eLB multiplexers are kept fixed to the worst case scenario of highest flexibility to clearly observe the effect of each customization of SB-R.

2- Silicon Properties

The silicon properties (ST65nm LP-SVT) of all the above mentioned custom tiles are described in table 4.7 for a LUT size 4, cluster size 4 and channel size 48 (represented as tsr_Ls Cs Chs, tsr_4 4 48 for instant readability of fundamental properties). The table shows details of area and configuration breakdowns of major components of the tile. The logic density (LUTs/mm^2), the static power and pessimistic dynamic power with default toggle rates and static probability values of Design Compiler. It is interesting to note the silicon properties according to the discussions of the seven tiles above. We can see that tsr_noNE and tsr_50pAll have same area; similarly all the tiles of theme 2 are architecturally equivalent from silicon point of view. Figure 4.21 and 4.22 graphically illustrates the properties of area and logic density showing how silicon efficient the custom tiles are compared to base tsr_F.

3- Architectural analysis (benchmarks mapping)

Finally we implemented all the benchmarks of table 4.6 on all the custom architectures, the results are illustrated in table 4.8 to table 4.10. Several parameters were analyzed to judge different customizations. The most important one is the maximum channel width used and critical path delay. The experiments are different also in a way that they are done in more practical and real-life like scenario. Instead of iterating for every benchmark individually to find minimum channel width (common in literature) all benchmarks are treated equally and mapped on a fixed architecture. The analyzers (chapter 3) gives the maximum channel used by each benchmark (provides roughly the equivalent of minimum channel width, see appendix A2 for more details) this helps to compare effect of different customizations as evident in the tables. The routing iteration tells about increased routing efforts/time tradeoff due to reduced flexibility of customization. The average channel width directly does not give much significant clue but is interesting to visualize how underused the channels are on average for any real scenario (was discussed it in section 4.1 also). By observing the results we can see that tsr_50pall and tsr_c1 performed poorly compared to others. To further validate it we repeated all experiments on channel size of 36 (provide higher pressure on router) which are shown in table 4.11 to 4.13 proving the observation (see Appendix A2 for silicon statistics).

If we analyze table 4.11 to 4.13 further in the light of figure 4.20 and figure 4.21 some interesting observations can be made. The tsr_noN and tsr_noNE both yield similar results, so the two-sided routing in the raw form is an interesting and more silicon efficient choice. The results of tsr_50pAll are surprising compared to tsr_noNE; it might be that cutting the 50% of all ports yield higher routing constraint compared to tsr_noNE as none of the channel can be fully exploited (plus there is no logical equivalence due to removed LI). The theme 2 except tsr_c1 yields the best silicon efficient and routing efficient (with very low tradeoff) architectures. We can clearly see the benefit of diversity here which distinguishes these experiments from several past works (CB based). It is interesting to note that tsr_c1 created the classical VPR-like architectural scenario where each LB pin is connected to one side of channels, due to lack of logical equivalence exploitation in this case due to eliminated LI, routing was severely constrained. The diversity of tsr_c2 and tsr_c3 almost nullified that problem yielding good routing efficiency despite no logical equivalence of eLB outputs. Figures 4.23 to 4.25 highlight the key outcomes of experiments, showing improved silicon efficiency (speed in particular), with negligible tradeoffs in routing efficiency.

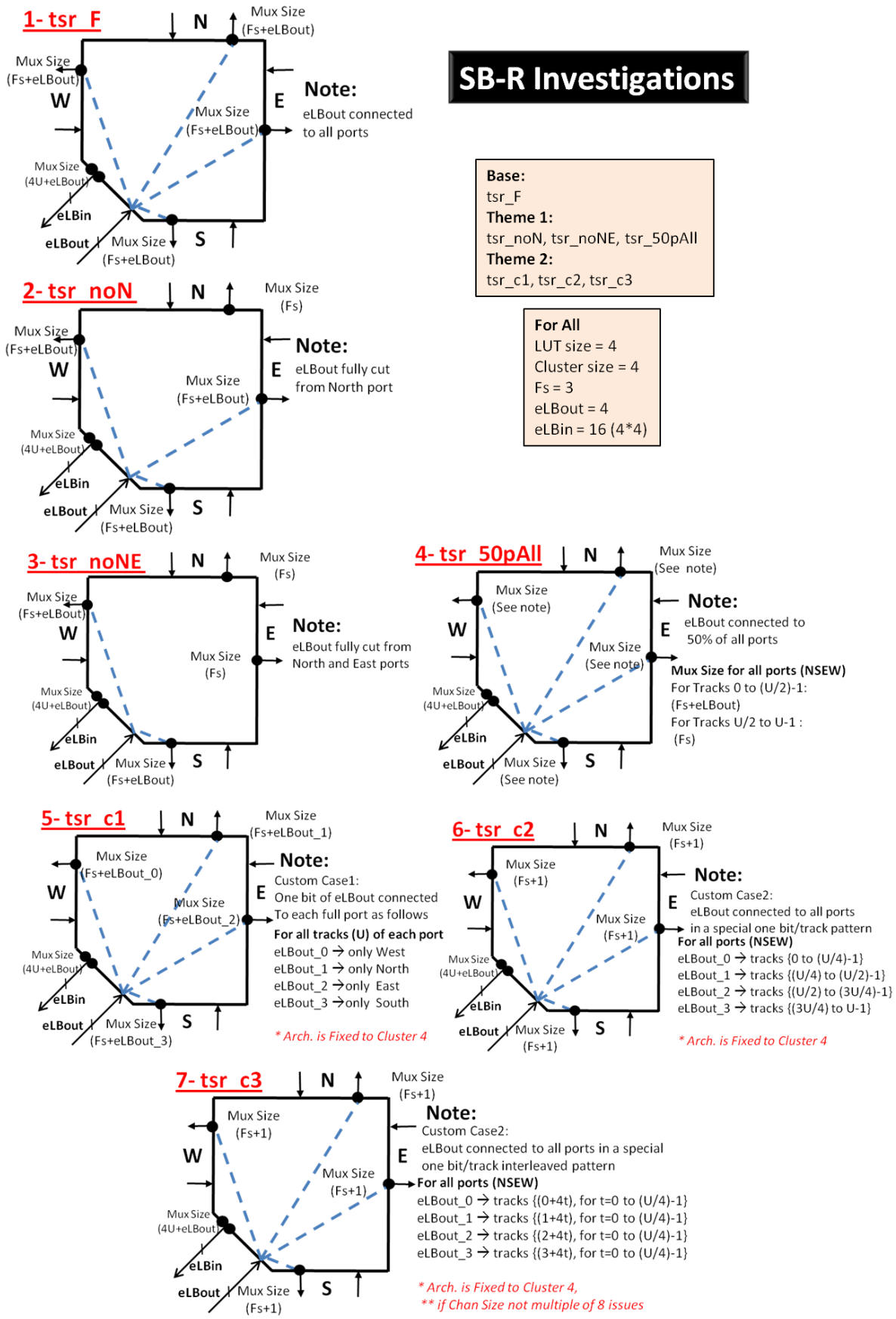


Fig. 4.20: The series of SB-Routing mux customization

		Base	Theme 1			Theme 2		
ST 65LPSVT 1.3V-m40C		tsr_4448_F	tsr_4448_noN	tsr_4448_noNE	tsr_4448_50pAll	tsr_4448_c1	tsr_4448_c2	tsr_4448_c3
Conf. bits	Total	468	444	420	420	372	372	372
	SBOX	400	376	352	352	304	304	304
	eLB	68	68	68	68	68	68	68
Area (um²)	Total	15023	14361	13794	13794	12740	12740	12740
	SB-R	2440	2064	1697	1697	1211	1211	1211
	SB-eLB	6720	6720	6720	6720	6720	6720	6720
	eLB	418	418	418	418	418	418	418
	Conf.	4043	3825	3686	3686	3257	3257	3257
	Buff.	1402	1334	1273	1273	1134	1134	1134
Logic Density (LUT4/mm²)		266	279	290	290	314	314	314
Area Normalized to base arch. F		1.00	0.96	0.92	0.92	0.85	0.85	0.85
Power	Static (nW)	452	433	417	417	385	385	385
	Dyn.* (uW/MHz)	3.67	3.57	3.33	3.32	3.06	3.04	3.06

Table 4.7: Silicon properties of 7 custom routing mux tiles for channel size 48 (*pessimistic power)

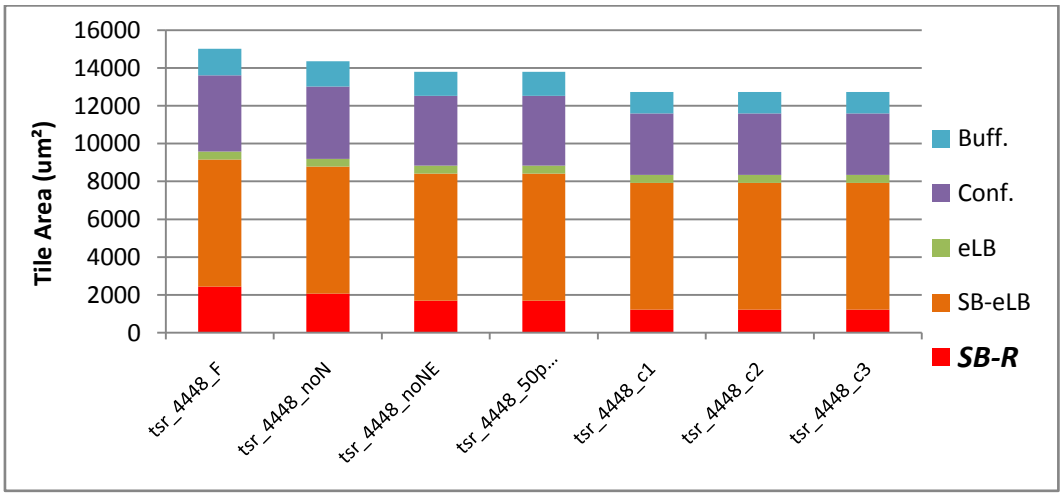


Fig. 4.21: Area breakdown for different SB-R customizations

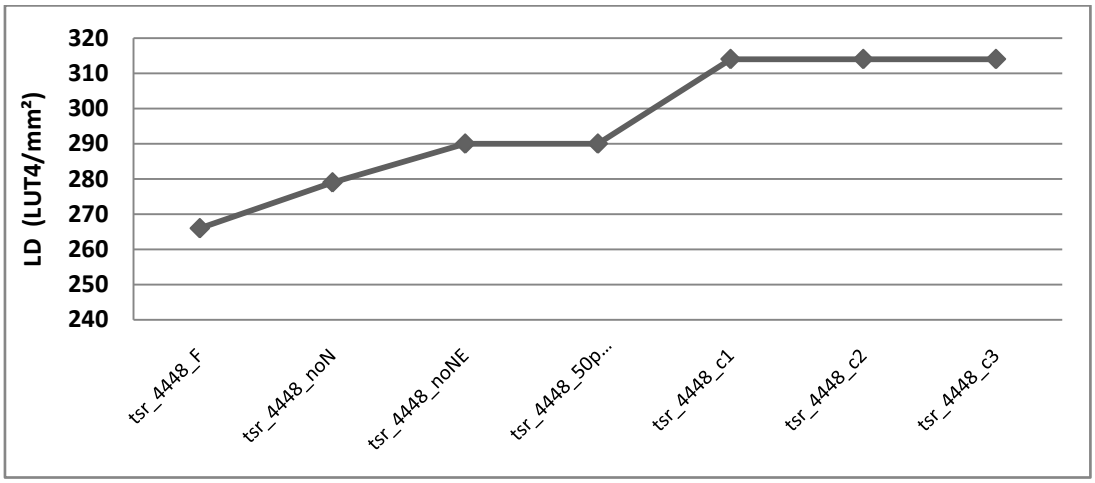


Fig. 4.22: Logic Density for different SB-R customized architectures

Circuit	tsr_4448_F			
	Ch. Max	Ch. Avg	R. Iter	Delay(ns)
ex5p	20	4.3	5	11.77
tseng	20	8.4	5	13.6
apex4	35	16.4	6	21
misex3	25	8.3	5	13.1
alu4	29	11.5	6	16.41
diffeq	21	8.9	5	15.64
seq	33	15.4	6	16.7
apex2	36	14.17	7	18.4
k2	30	11.1	5	17.28
s298	36	17.3	7	23.87
ex1010	35	18.7	8	20.45
Arith. Mean	**29.1**	**12.22**	**5.91**	**17.11**

Table 4.8: PAR results for tile tsr_4448_F

Circuit	tsr_4448_noN				tsr_4448_noNE				tsr_4448_50pAll			
	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)
ex5p	23	4.7	5	10.42	21	5.2	5	10.22	20	4.5	5	10.31
tseng	22	8.9	5	14.58	22	9.5	5	12.1	20	8.4	6	13.1
apex4	35	16.6	7	18.12	35	17.3	8	16.1	RF			
misex3	26	8.4	6	11.57	28	9	5	11.55	22	8.4	7	11.53
alu4	31	11.7	6	14.59	29	12.3	7	13.72	26	11.7	7	16.28
diffeq	25	9.4	6	14.89	27	9.8	6	14.14	22	9.3	7	15.74
seq	33	15.9	6	14.42	36	17.12	7	13.33	29	16.2	10	15.82
apex2	40	14.9	7	16.8	35	15.8	7	16.22	29	14.9	9	16.44
k2	30	11.5	7	13.48	32	12.4	7	12.51	25	11.8	9	14.29
s298	38	18	7	22.16	41	19.42	7	18.47	RF			
ex1010	37	19.32	7	17.95	36	20.5	7	16.36	RF			
Arith. Mean	**30.91**	**12.67**	**6.27**	**15.36**	**31.1**	**13.48**	**6.45**	**14.06**	**24.1**	**10.65**	**7.5**	**14.19**

Table 4.9: mapping results of tsr_4448 theme1 tiles

Circuit	tsr_4448_c1				tsr_4448_c2				tsr_4448_c3			
	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)
ex5p	24	5.4	6	9.37	18	4.4	4	8.1	19	4.4	5	8.38
tseng	RF				20	8.4	6	11.2	20	8.4	6	10.6
apex4	RF				38	15.8	6	14.6	35	15.7	7	13.58
misex3	30	9.5	6	10.07	26	7.87	6	9.28	27	8	5	9.37
alu4	31	13.2	6	13.05	29	11.3	6	11.67	29	11.3	5	11.43
diffeq	RF				26	9	5	11.62	23	9	6	11.65
seq	RF				33	15	7	11.41	32	15	7	12.43
apex2	RF				31	14.1	7	13.91	32	14	7	13.69
k2	36	13.3	6	13	28	11.2	7	11.13	31	11	7	10.19
s298	RF				36	16.6	7	18.45	34	17	7	17.1
ex1010	RF				37	18.5	14	14.02	35	18.4	13	14.25
Arith. Mean	**30.25**	**10.35**	**6**	**11.37**	**29.3**	**12**	**6.82**	**12.31**	**28.82**	**12.02**	**6.8**	**12.06**

Table 4.10: mapping results of tsr_4448 theme2 tiles

Circuit	tsr_4436_F			
	Ch. Max	Ch. Avg	R. Iter	Delay(ns)
ex5p	21	4.49	5	12.85
tseng	21	8.4	7	14.08
apex4	33	16.83	8	20.22
misex3	25	8.1	6	12.36
alu4	26	11.84	7	16.41
diffeq	22	9.1	6	16
seq	32	15.98	8	15.86
apex2	30	15.24	7	18.57
k2	29	11.38	6	15.61
s298	32	17.84	9	24.94
ex1010	33	19.45	8	20.27
A. Mean	**27.64**	**12.6**	**7**	**17.02**

Table 4.11: PAR results for tile tsr_4436_F

Circuit	tsr_4436_noN				tsr_4436_noNE				tsr_4436_50pAll			
	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)
ex5p	22	4.8	5	11.84	21	5.2	5	10.2	19	4.6	6	11.8
tseng	24	8.9	5	12.81	20	9.7	6	12.29	19	8.5	7	12.7
apex4	33	17.5	8	18.2	33	17.8	9	16.2	RF			
misex3	26	8.6	6	12.1	29	9.4	6	11.1	22	8.4	9	13.3
alu4	29	11.8	6	14.2	29	12.7	7	13.3	22	12.3	60	14.66
diffeq	23	9.3	7	15	25	9.8	6	14.11	18	9.4	14	16.39
seq	30	16.5	7	14.6	33	17.5	7	13	RF			
apex2	32	16	8	16.6	32	17	8	15.58	RF			
k2	30	12	6	14.44	31	12.9	7	12.33	RF			
s298	36	18.3	8	22.97	33	19.8	11	20.9	RF			
ex1010	34	20.5	10	17.3	34	21.2	20	17.4	RF			
Arith. Mean	**29**	**13.1**	**6.9**	**15.46**	**29.1**	**13.9**	**8.4**	**14.22**	**20**	**8.64**	**19.2**	**13.77**

Table 4.12: mapping results of tsr_4436 theme1 tiles

Circuit	tsr_4436_c1				tsr_4436_c2				tsr_4436_c3			
	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)
ex5p	25	5.73	5	9.75	20	4.6	6	8.98	22	4.48	5	8.8
tseng	RF				20	8.3	6	10.83	20	8.3	5	11.03
apex4	RF				RF				RF			
misex3	26	9.8	6	10.37	25	8.2	8	9.4	23	8.2	7	9.59
alu4	30	13.7	7	12.35	27	11.3	10	11.46	26	11.4	10	11.64
diffeq	RF				22	8.96	7	12.84	24	8.78	6	12.87
seq	RF				31	15.6	8	12.27	30	15.8	8	11.57
apex2	RF				33	14.9	7	14.16	33	14.7	7	13.85
k2	32	14	7	12.07	28	11.6	8	11.83	29	11.5	7	11.8
s298	RF				RF				RF			
ex1010	RF				RF				RF			
Arith. Mean	**28.3**	**10.8**	**6.25**	**11.14**	**25.8**	**10.43**	**7.5**	**11.47**	**25.9**	**10.4**	**6.9**	**11.39**

Table 4.13: mapping results of tsr_4436 theme2 tiles

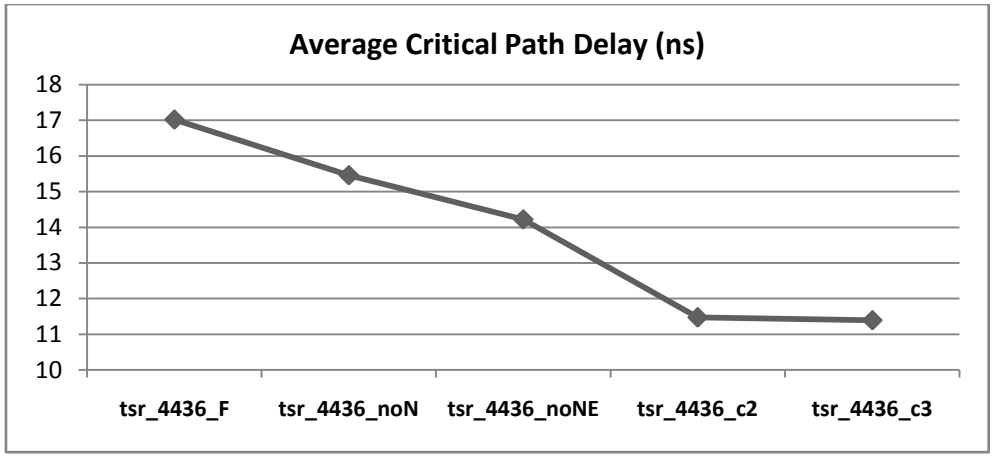


Fig. 4.23: Average critical path delay of benchmarks for different SB-R customizations

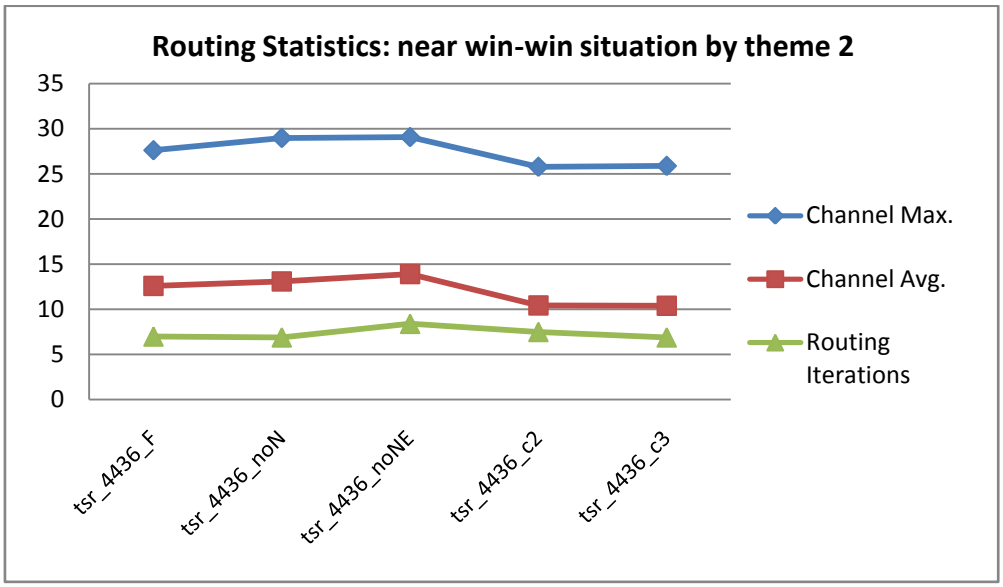


Fig. 4.24: Average routing statistics of benchmarks for different SB-R customizations

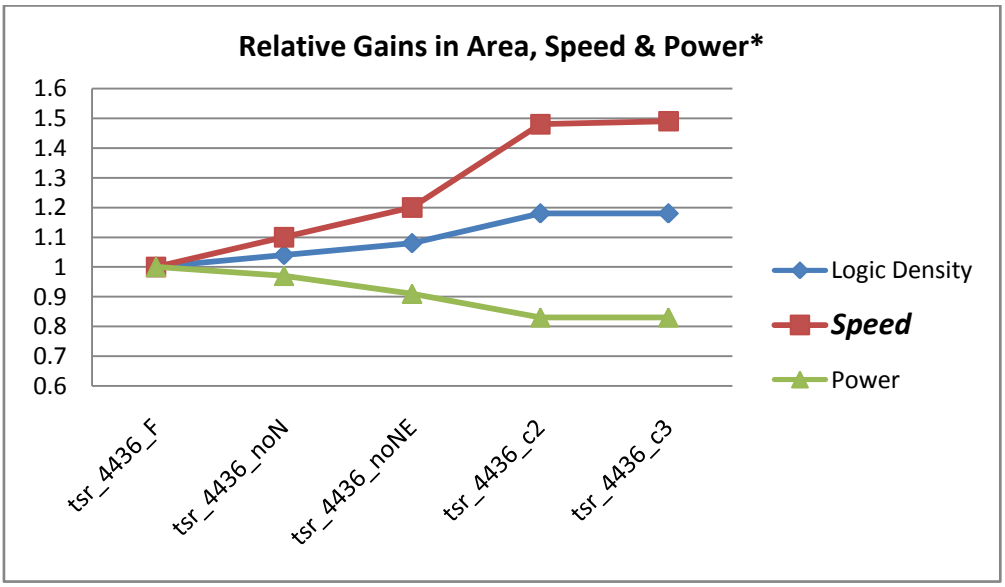


Fig. 4.25: Silicon tradeoffs for different SB-R customizations (*pessimistic power)

4.2.3 SB-eLB Interconnect Multiplexers optimization

In a similar fashion and motivation like the previous section, this section details the explorations of different customization ways of SB-eLB multiplexers optimization. It first describes the experimented custom architectures properties and then the silicon implementation and benchmark mappings results are presented in detail.

1- Custom Architecture types

The SB-eLB multiplexers (figure 4.19) are the routing multiplexers which allow switching of different routing tracks to the inputs of eLBs or in other words connect eLB inputs to the routing network. These multiplexers are lesser in quantity compared to SB-R muxes but much larger in size (figure 4.19). The objective is to reduce the size of these multiplexers in such a way that more silicon efficient (area, power, speed) architecture is created with minimal tradeoff to the routing efficiency for mapped applications. For experimentation seven distinct types of customizations were investigated in a similar fashion like SB-R explorations of previous section.

They can be divided in three themes similar to SB-R explorations which are discussed below and shown in figure 4.26 in detail with their properties. The feedbacks (eLB outputs) are fully connected to eLB inputs.

Base (Full)

tse_F: Exactly similar like tsr_F of previous section. It provides the base case for reference with highest routing flexibility (worst silicon efficiency). It represents worst case of figure 4.19.

Theme/Strategy 1 (cutting ports partly/fully)

The customizations of this theme are partly inspired by our state of art discussions in chapter 2 (figure 2.10) for routing architecture sides. It addresses the effects of cutting eLB inputs partly/fully from ports.

tse_noN: The eLB inputs are fully cut from traffic of North (N) port, {three-sided routing}.

tse_noNE: The eLB inputs are fully cut from North (N) and East (E) port traffic, {two-sided routing}.

tse_50pAll: The eLB inputs are connected to 50% of all four ports of SB. In principle it is architecturally equivalent to tse_noNE, instead of fully cutting two ports 50% of all four are used.

Theme/Strategy 2 (diverse customization)

The customizations of this theme address diverse customization in different forms in such a way that every eLB input is connected to an equivalent of a single port only (four times optimal than the base case) yielding a highly silicon efficient solution particularly in terms of area (since SB-eLB muxes are very large as we noticed also in section 4.1 silicon discussions).

tse_c1: Each eLB input in a repetitive order as LUT input in cluster is diversely connected to all four ports in a manner shown mathematically in the figure 4.26.

tse_c2: In a similar fashion like tse_c1 the eLB inputs are diversely connected in more interleaved form to all four ports in the mathematical manner shown in figure 4.26.

tse_c3: Each eLB input in a repetitive order as LUT inputs in cluster is connected to a single port. This represents a near similar classical scenario of VPR-like CB based architecture where each pin is connected to a single side of routing through the CB (relatively more layout friendly way for full custom).

For all the cases in the seven custom tiles the SB-R multiplexers are kept fixed to the worst case scenario of highest flexibility to clearly observe the effect of each customization of SB-eLB. Furthermore eLB outputs are fully connected to each eLB input in these experiments providing full flexibility.

2- Silicon Properties

The silicon properties (ST65nm LP-SVT) of all the above mentioned custom tiles are described in table 4.14 for a LUT size 4, cluster size 4 and channel size 48 (represented as tse_Ls Cs Chs, tsr_4 4 48 for instant readability of fundamental properties). The table shows details of area and configuration breakdowns of major components of the tile. The logic density (LUTs/mm^2), the static power and pessimistic dynamic power with default toggle rates and static probability values of Design Compiler. It is interesting to note the silicon properties according to the discussions of the seven tiles above. We can see that tse_noNE and tse_50pAll have same area; similarly all the tiles of theme 2 are architecturally equivalent from silicon point of view. Furthermore we can also observe the configuration size abrupt change issue partly here (in a slightly different scenario). Moving from tse_F to tse_noN we observe that the area goes down obviously but configuration size is same (will not always be the case) due to large multiplexer sizes of SB-eLB, when we go higher in mux size the addressable range dramatically changes due to obvious reasons (e.g conf. size of 7 can select from 65 to 128 signals). Figure 4.27 and 4.28 graphically illustrates the properties of area and logic density showing how silicon efficient the custom tiles are compared to base tsr_F. Compared to SB-R investigation here we see dramatic improvements in area due to high gains of SB-eLB mux size customization (particularly for theme 2) as shown in graphs.

3- Architectural analysis (benchmarks mapping)

Finally we implemented all the benchmarks of table 4.6 on all the custom architectures, the results are illustrated in table 4.15 to table 4.17. In a similar fashion and motivations described in SB-R explorations we observed the effects of customization on channel demands, routing iteration and delay of critical path.

If we analyze table 4.15 to 4.17 further in the light of architecture properties (figure 4.26), silicon properties (table 4.14 and figure 4.27) and knowledge gained from SB-R investigations some interesting observations can be made. In a similar manner like was for SB-R the tse_noN and tse_noNE both yield similar results, so the two-sided routing in the raw form is an interesting and more silicon efficient choice. The results of tse_50pAll are poor in this case too. Theme 2 (except tse_c3) provides the best silicon efficient architectures with acceptable tradeoff in terms of channel demand. As we observed poor results for the SB-R case of tsr_c1 similar scenario was observed in SB-eLB for an equivalent case of tse_c3 in which we distributed the eLB inputs in a classical CB based VPR-like architecture style to unique ports. Due to lack of logical equivalence (eliminated LI) poor results were obtained. The diverse distribution of tse_c1 and tse_c2 greatly nullified that problem. It is important to note that for SB-eLB experiments the natural logical equivalence of eLB inputs in a marginal/group form as LUT (BLE) inputs was not exploited as it was not fully supported by the version of eFPGA Programmer which was used for experiments. That feature is almost universal in all commercial FPGAs (adaptable LUT configuration) and assumed in all research works (VPR etc.). Revising these experiments with this marginal equivalence exploitation will further improve the presented results.

Figures 4.29 to 4.31 illustrate the key aspects of research findings of table 4.15 to 4.17. It is interesting to note that for the case of SB-eLB explorations we achieved a significant gain for the area (logic density) but for the critical path delay the effect was nearly flat (to better illustrate, in the graphs compared to tables we took the average of each case by ignoring last two benchmarks). The reason is that often the critical path passes to several SBs in its way to reach other tiles. Hence multiple SB-R crossings dominate the total delay compared to fewer SB-eLB crossings. The physical long wires and improved routing algorithms can further help and better exploit more silicon efficient architectures. Nevertheless with SB-R and SB-eLB explorations we have greatly improved the eFPGA architectural efficiency with negligible tradeoff on routing efficiency (win-win situation) compared to base architectures which we explored in section 4.1.

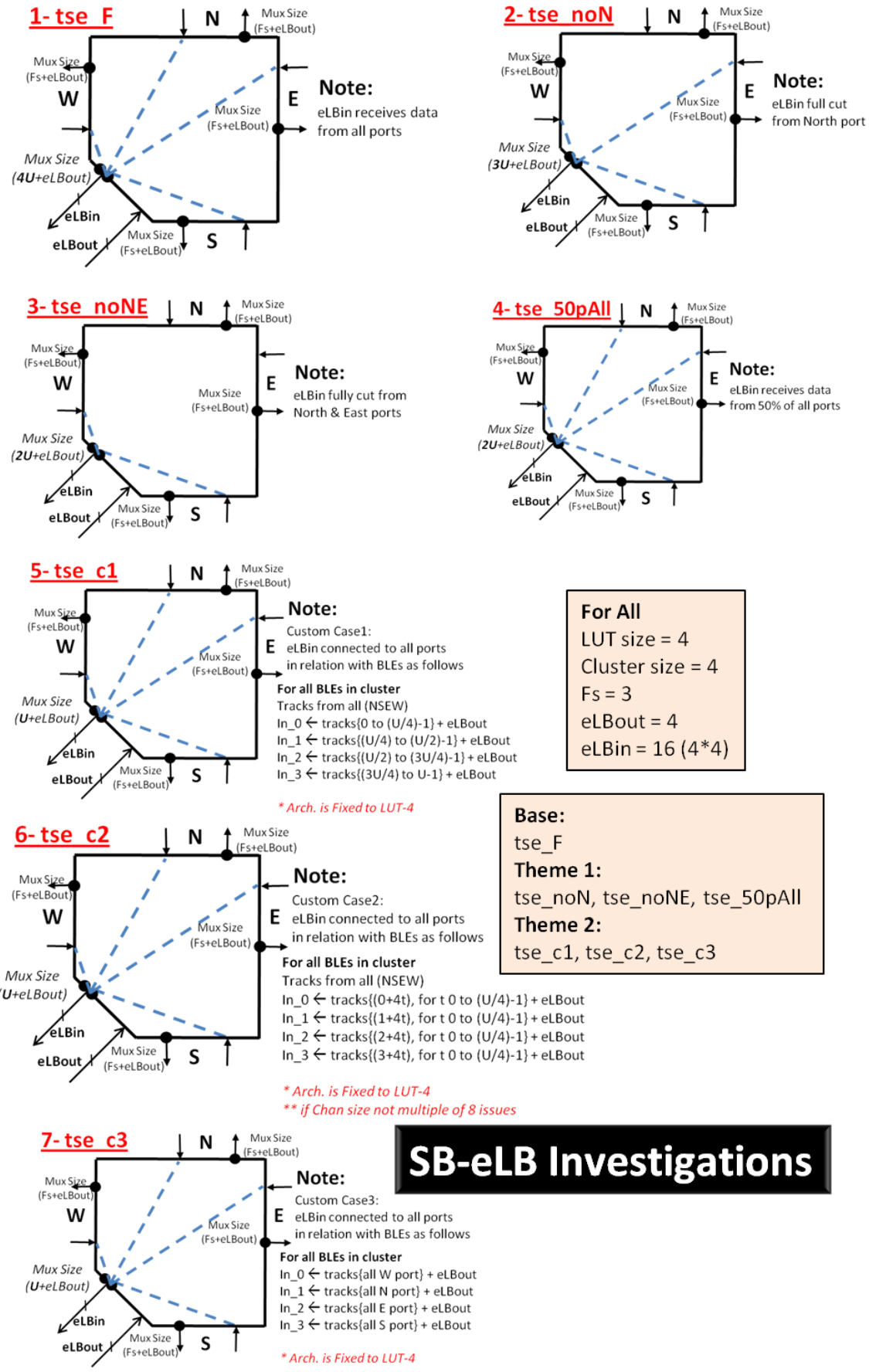


Fig. 4.26: The SB-eLB multiplexers optimizations experiment custom architectures

		Base	Theme 1			Theme 2		
ST 65LPSVT 1.3V-m40C		tse_4448_F	tse_444_noN	tse_4448_noNE	tse_4448_50pAll	tse_4448_c1	tse_4448_c2	tse_4448_c3
Conf. bits	Total	468	468	452	452	436	436	436
	SBOX	400	400	384	384	368	368	368
	eLB	68	68	68	68	68	68	68
Area (um²)	Total	15023	13550	11769	11758	9881	9881	9881
	SB-R	2440	2440	2440	2440	2440	2440	2440
	SB-eLB	6720	5232	3600	3600	1984	1984	1984
	eLB	418	418	418	418	418	418	418
	Conf.	4043	4066	3960	3960	3810	3810	3810
	Buff.	1402	1394	1351	1340	1229	1229	1229
Logic Density (LUT4/mm²)		266	295	340	340	405	405	405
Area Normalized to base arch. F		1.00	0.90	0.78	0.78	0.66	0.66	0.66
Power	Static (nW)	452	415	365	365	304	304	304
	Dyn.* (uW/MHz)	3.67	3.45	2.87	2.91	2.44	2.41	2.42

Table 4.14: Silicon properties of 7 custom SB-eLB mux tiles for channel size 48 (*pessimistic power)

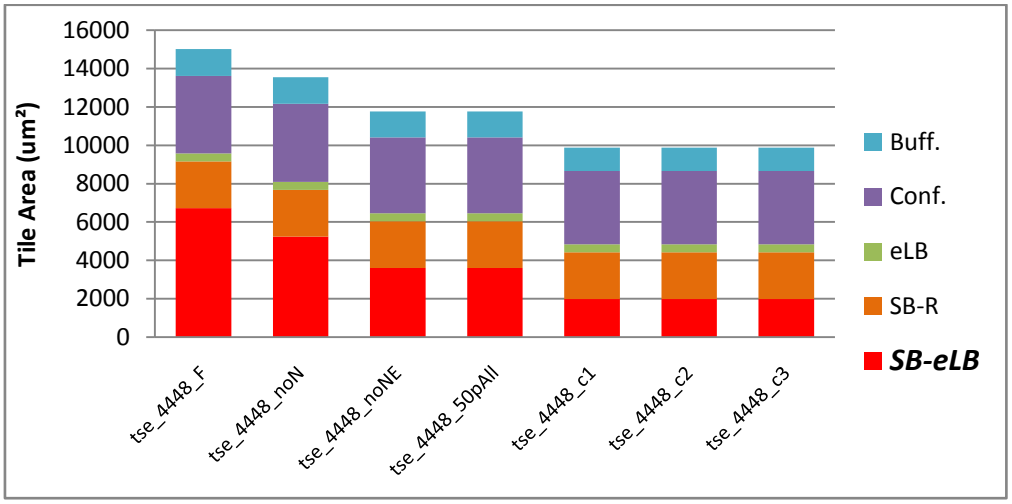


Fig. 4.27: Area breakdown for different SB-eLB customizations

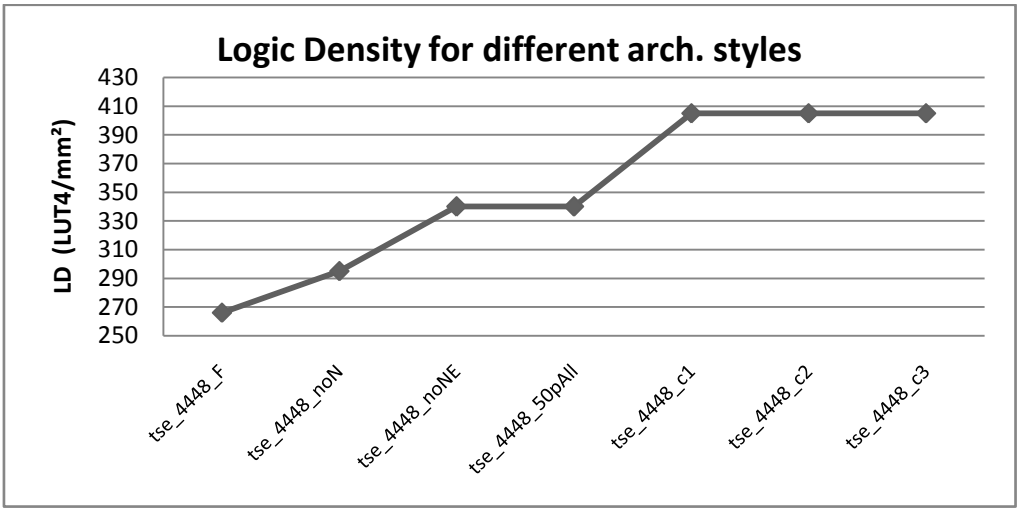


Fig. 4.28: Logic Density for different SB-eLB customized architectures

Circuit	tse_4448_F			
	Ch. Max	Ch. Avg	R. Iter	Delay(ns)
ex5p	20	4.27	5	11.77
tseng	20	8.4	5	13.6
apex4	35	16.4	6	20.99
misex3	25	8.26	5	13.1
alu4	29	11.49	6	16.41
diffeq	21	8.93	5	15.64
seq	33	15.37	6	16.7
apex2	36	14.2	7	18.38
k2	30	11.1	5	17.28
s298	36	17.3	7	23.87
ex1010	35	18.7	8	20.45
Arith. Mean	**29.1**	**12.22**	**5.91**	**17.11**

Table 4.15: PAR results for tile tse_4448_F

Circuit	tse_4448_noN				tse_4448_noNE				tse_4448_50pAll			
	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)
ex5p	23	5	5	11.65	27	5.72	6	12.92	18	4.54	6	11.88
tseng	28	9.2	6	13.44	24	11	7	14.64	20	8.6	6	14.12
apex4	36	17.6	7	20.72	39	20.25	7	19.96	RF			
misex3	28	8.9	6	12.73	31	10.4	8	13.3	23	8.5	7	12.54
alu4	31	12.3	7	17.8	33	14.3	8	17.73	24	12	10	16.98
diffeq	24	9.6	7	15.93	29	11.7	7	17.2	19	9	7	15.6
seq	37	16.9	7	16.13	37	19.2	7	15.84	RF			
apex2	34	15.6	7	19.35	38	18.8	8	19.23	24	16.4	94	22.84
k2	33	12.7	7	14.06	36	14.4	7	16.56	25	13.4	55	18.57
s298	41	19.1	7	24.5	41	21.3	8	28.06	RF			
ex1010	38	20.7	8	21.06	41	24.5	8	21.98	RF			
Arith. Mean	**32.1**	**13.4**	**6.7**	**17.03**	**34.2**	**15.6**	**7.4**	**17.95**	**21.86**	**10.3**	**26.43**	**16.08**

Table 4.16: mapping results of tse_4448 theme1 tiles

Circuit	tse_4448_c1				tse_4448_c2				tse_4448_c3			
	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)	Ch. Max	Ch. Avg	R. Iter	Delay (ns)
ex5p	30	6.5	6	10.9	30	6.4	6	12.01	32	7.3	6	12.8
tseng	25	9.4	5	12.76	27	9.4	5	13.1	RF			
apex4	42	23.5	10	20.63	41	23.5	24	21.2	RF			
misex3	34	12.3	6	12.25	33	12.2	7	11.2	33	13.6	8	13.41
alu4	36	16.9	7	15.46	38	16.7	7	17.2	37	18.2	7	16.74
diffeq	27	10.4	6	15.34	26	10.7	6	15.34	RF			
seq	39	23.5	15	15.92	43	23	12	15.3	RF			
apex2	41	22.36	12	17.85	40	22.5	10	19.23	RF			
k2	38	15.8	7	15.4	39	15.78	8	15.63	39	17.6	8	15.9
s298	RF				RF				RF			
ex1010	RF				RF				RF			
Arith. Mean	**34.7**	**15.63**	**8.22**	**15.17**	**35.2**	**15.58**	**9.4**	**15.58**	**35.25**	**14.18**	**7.25**	**14.7**

Table 4.17: mapping results of tse_4448 theme2 tiles

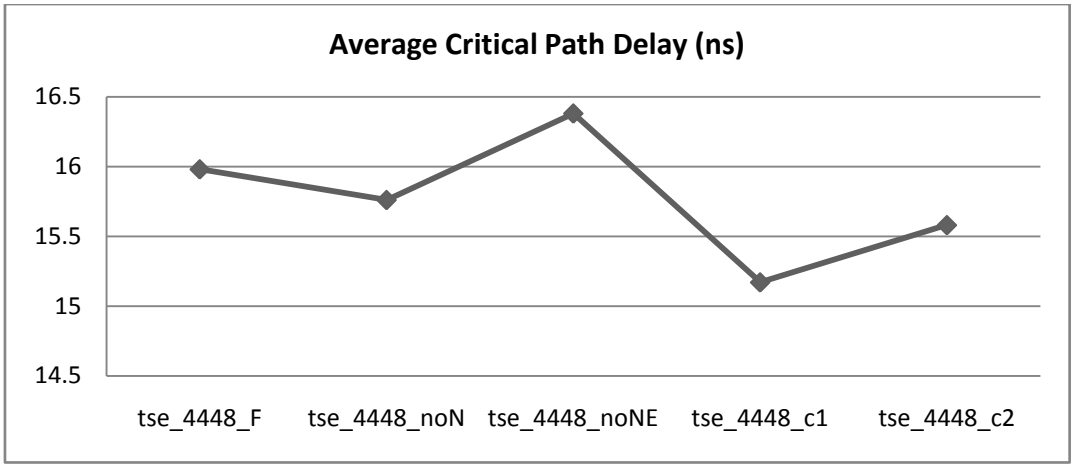


Fig. 4.29: Average critical path delay of benchmarks for different SB-R customizations

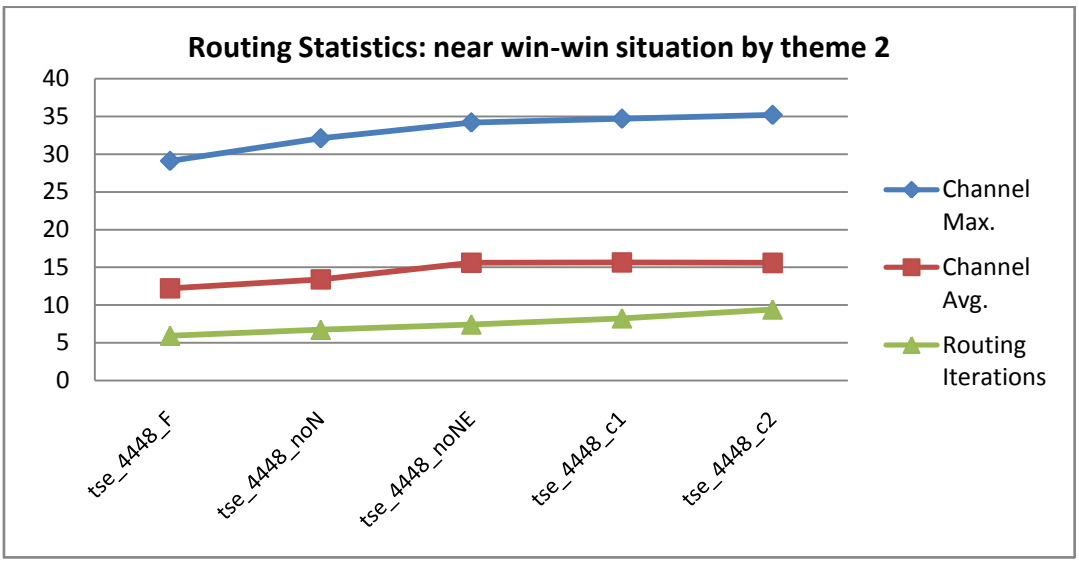


Fig. 4.30: Average routing statistics of benchmarks for different SB-eLB customizations

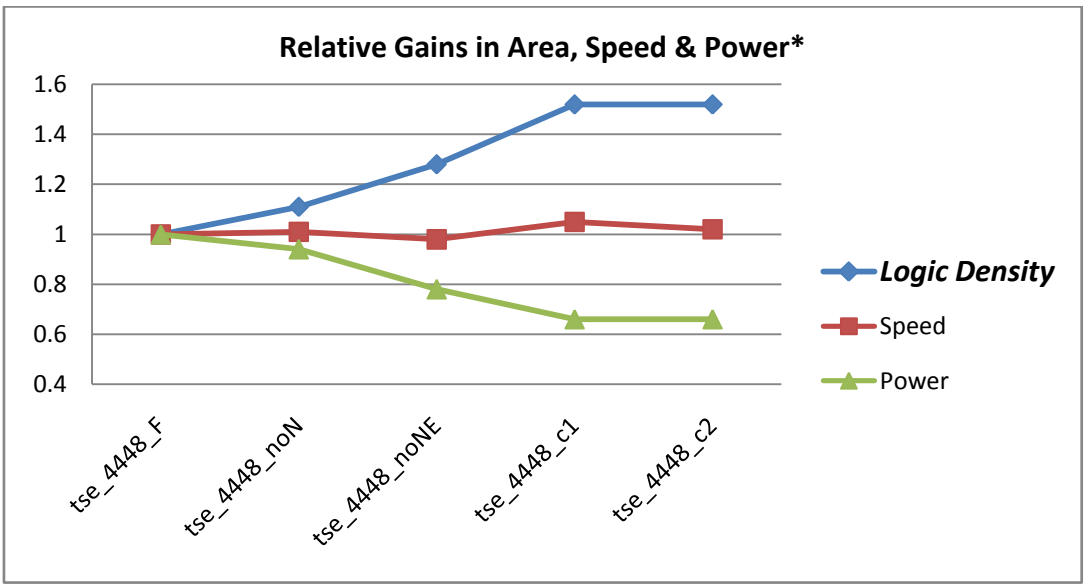


Fig. 4.31: Silicon tradeoffs for different SB-eLB customizations (*pessimistic power)

4.2.4 Combined optimizations and explorations

This section provides general comparison of the results achieved in SB-R and SB-eLB customization investigations in the previous sections with the classical architecture published results (VPR-like). It investigates merging of the SB-R and SB-eLB explorations to find joint optimal architecture for these investigations. A brief overview of the process node parameters variation like threshold voltage, process type etc. on power vs speed on the best found architecture is also presented.

1- General Comparison with classical architecture

In the previous sections we extensively explored different dimensions of customization of SB-R and SB-eLB multiplexers to find silicon efficient architectures with only marginal tradeoff in routing efficiency due to decreased flexibility. It will be interesting to revisit figure 4.18 and make some comparisons with published results of classical architecture to find what we have achieved through these new experiments and what pros and cons were observed. Although full direct comparison (silicon+architecure) is relatively difficult to make since we are not investigating layout based architectures. However the relative comparison of logical architecture (in terms of parameters like Fc etc.) can be roughly made to get an idea. [Marquardat.Betz.Rose00, 4.10] has investigated the tradeoffs of cluster based FPGAs (VPR) and found that for MCNC benchmarks the best value for Fc,in for cluster size of 4 (our case) is 0.6W and W/N for Fc,out. According to our explored investigations in previous sections we found the optimal values of SB-R multiplexers to be Fs+1 (tsr_c2 and tsr_c3 topologies) and for SB-eLB multiplexers to be U+N (tse_c1 and tse_c2 topologies), however they were explored and found separately (next section will address that). Figure 4.32 illustrates these results in connection with figure 4.18. We can clearly observe that these new experiments by exploiting the diverse form of connectivity of logic block (eLB) inputs and outputs to the routing network through unified SB has led to attractive results which rival the widely addressed classical architecture with connection block and fully connected local interconnect. It is worth noticing that the experiments that we conducted were not exploiting the natural logical equivalence of LUT inputs, exploiting that will further help improving results which are already attractive. The next section explores the combination of best results of previous two sections to investigate how well they perform together.

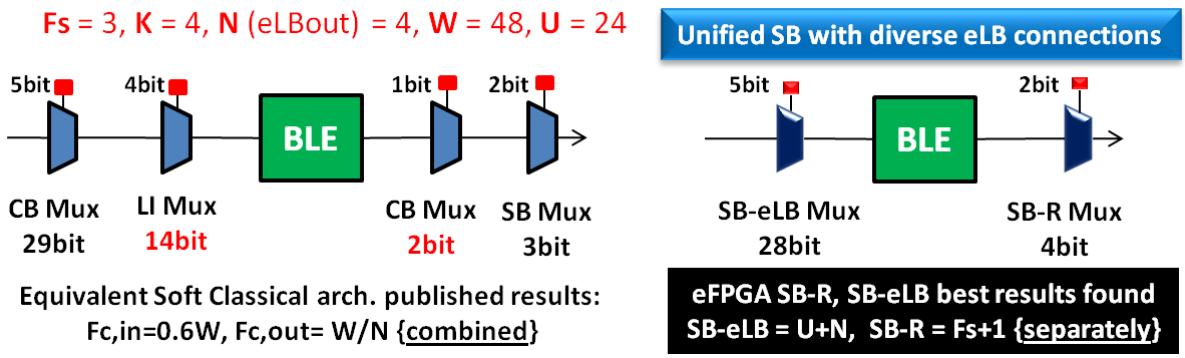


Fig. 4.32: Architectural results comparison between classical (CB+LI) and unified SB eFPGA

2- Combined experiments for SB-R+SB-eLB joint best cases

In the previous section we observed high architectural efficiency (figure 4.32) achieved through unified SB with diverse connectivity of eLB inputs and outputs to the routing network by independently exploring SB-R and SB-eLB customizations. This section investigates merging of the explorations to find optimal benefits of both cases. We observed during explorations, that SB-R customizations led to significant improvements in speed (figure 4.25), while SB-eLB was yielding significant improvements for logic density (figure 4.31). The joint explorations were conducted in six different cases divided into three themes which are described below. For simplicity detailed exploration results for benchmarks mapping and silicon statistics of tiles are not presented, they can be found in Appendix A2.

Theme 1 (Combining the best cases of SB-R and SB-eLB)

In the first theme, the best cases of SB-R (tsr_c2, tsr_c3) and SB-eLB (tse_c1, tse_c2) were explored in the form of two cases which are shown below. *They failed to map any benchmark application*, the reduced flexibility of the best cases combined was too-restrictive for router to route the benchmarks. It will be interesting for future investigations to better understand reasons and innovations to address that.

Case 1: SB-R c2 + SB-eLB c1 ; Case 2: SB-R c3 + SB-eLBc2

Theme 2 (Combining second best of SB-R and best of SB-eLB)

From the knowledge of unsuccessful behavior of theme 1, the degree of hardware routing flexibility was increased in hybrid way. The second best case of SB-R exploration that was two ports elimination (tsr_noNE) was selected as SB-R and the best cases of SB-eLB (tse_c1, tse_c2). The two explored cases are shown below. Unlike theme 1 theme 2 explorations succeeded to map benchmarks effectively, the detailed exploration results can be seen in appendix A2.

Case 3: SB-R noNE + SB-eLB c1 ; Case 4: SB-R noNE + SB-eLB c2

Theme 3 (Combining best of SB-R and second best of SB-eLB)

In a similar fashion like theme 2, in theme 3 the best case of SB-R (tsr_c2, tsr_c3) and second best case of SB-eLB with two ports elimination (tse_noNE) was used. The experiments were successful like theme 2. The details of experiments can be found in appendix A2.

Case 5: SB-R c2 + SB-eLB noNE ; Case 6: SR-Rc3+SB-eLB noNE

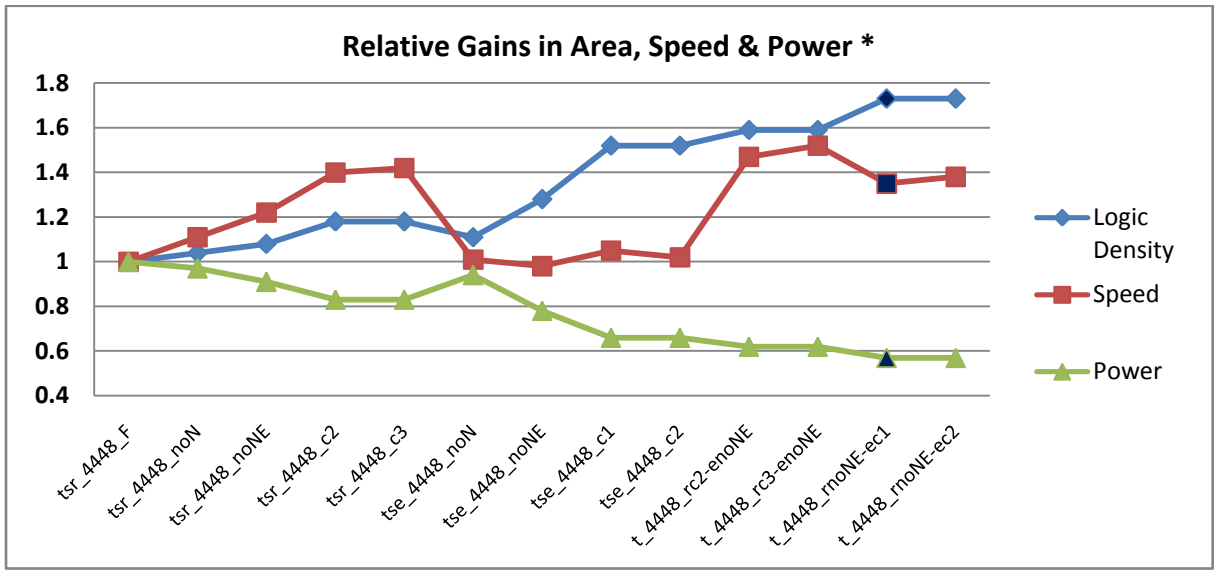


Fig. 4.33: Detailed analysis of Silicon tradeoffs for all experiments (*pessimistic power)

Best overall Architectures (Golden Tiles)

The theme 2 and theme 3 yield the final best optimal architectures for the explorations done in section 4.2. Figure 4.33 graphically illustrates (relative to base tile which was achieved from section 4.1 basic experiments) the effects of customizations on logic density (area), speed (average critical path delay of benchmarks) and power (pessimistic power). We can clearly observe the independent effect of SB-R innovations (tsr), SB-eLB innovations (tse) and the combined effect of theme 2 (best for area) and theme 3 (best for speed). For theme 2 and theme 3 both the corresponding cases provided near similar result so

anyone can be chosen. So in conclusion of the long explorations, two architectures can be presented as the overall best (Golden Tiles). Their detailed silicon properties can be found in appendix A2.

- **t_4-4-X_rc2-enoNE** (Theme 3: Case 5): *Best for Speed*
- **t_4-4-X_rnoNE-ec1** (Theme 2: Case 3): *Best for Area and Power*

Based on target needs/constraints (area, speed, power etc.) the best eFPGA Core can be created with selecting one of these tiles pattern (LUT-4, Cluster-4) with suitable channel size (48 was used in presented experiments, can highly vary based on target applications as we discussed in section 4.1.4).

3- Effect of threshold and process node type (power vs speed)

We outlined the effect of threshold voltage on power and speed in discussions of power consumption challenges in section 4.1.5. This section provides a revisit to the discussions with extended experiments. We took the best tile (Theme 2: Case 3) from previous section and analyzed in detail the effect of threshold voltage and process type on power vs speed. The details of experiments are discussed below.

Table 4.18 shows the details of silicon statistics of the tile on various options of ST65nm process. What is further interesting in these experiments compared to the section 4.1.5 experiments is that in these experiments the GP (General Purpose) process is also experimented in addition to LP (Low Power) which was not available to us when we conducted the experiments of section 4.1 [S-9]. This allows observing the relative effect between LP and GP process on power and speed on same process node (we observed cross 90nm GP and 65nm LP comparison in figure 4.15). We can clearly observe the huge impact of threshold voltage and process type on static power. It is important to note that there is a slight difference in supply voltage (power has a square relation with voltage). We tried to make as best equivalent comparison as possible. For both cases the equivalent nominal case libraries for LP and GP were used. We can see the high variation of static power with the change of threshold voltage from low to high and despite a small difference in supply voltage, a phenomenal shoot in static power for GP process. The last row of the table highlights the magnitude of change in static power in relation with LPSVT process which we used throughout for majority of our experiments in this thesis work.

ST 65nm Process	**1.2V_25degreeC**			**1.0V_25degreeC**		
t_4448_rnoNE-ec1	**LPLVT**	**LPSVT**	**LPHVT**	**GPLVT**	**GPSVT**	**GPHVT**
Area (um^2)	8864	8676	8398	8662	8674	8687
Dynamic Power* (uW/MHz)	1.89	1.82	1.39	1.62	1.47	1.34
Static Power (nW)	8200	682	32	821000	118000	19000
Static Power relative to LPSVT	12	1	0.047	1204	173	27.86

Table 4.18: Silicon statistics of best tile (*pessimistic power)

Figure 4.34 graphically illustrates the results of table 4.18; we can observe static power from being negligible in LP to significant in GP, with exceptionally high for GPLVT. Figure 4.35 shows the relative critical path delay variation with respect to LPSVT process for the benchmark applications (detailed results can be found in appendix A2). We can clearly observe the power vs speed scenario. This also highlights how a same logical architecture can have multiple electrical implementations varied by process type with significant difference in power and speed for same area (notice in table 4.18, the area remains almost same for all types).

Benefits of Full custom FPGAs: We observed in chapter 2, how state of art FPGAs make complex use of different transistor types, threshold voltage etc. to combat power vs speed with best possible tradeoffs. This also highlights the benefits of full custom layout based FPGAs (commercial FPGAs are full custom as ROI justifies that), which implement the same logical architecture with different electrical options for implementation. Such things are relatively quite hard to exploit with soft eFPGAs (tradeoff of target independence), that is why the motivation of this work is small sized custom eFPGAs where silicon

tradeoff with ease of technology independence and customization is affordable and good value proposition. In practice, even for logical architecture components (LUTs, Multiplexers etc.), state of art use different kind of electrical architectures addressing some specific architecture challenges (e.g. some inputs of multiplexers faster than others, different topologies of multiplexers optimized for speed or area etc.) which can be analyzed in patents of state of art [3] (see patents survey of appendix A1 for details). [Kuon&Rose09, 5.2] provide good basics in this regard.

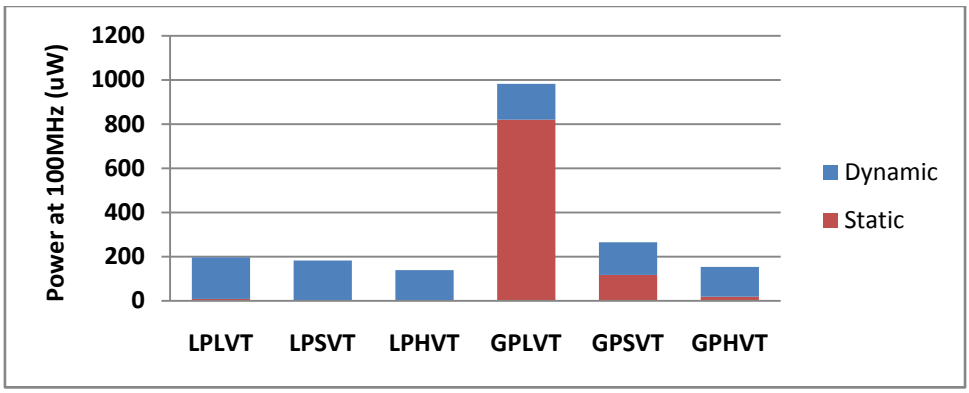


Fig. 4.34: Effect of threshold voltage and process type of same node on power

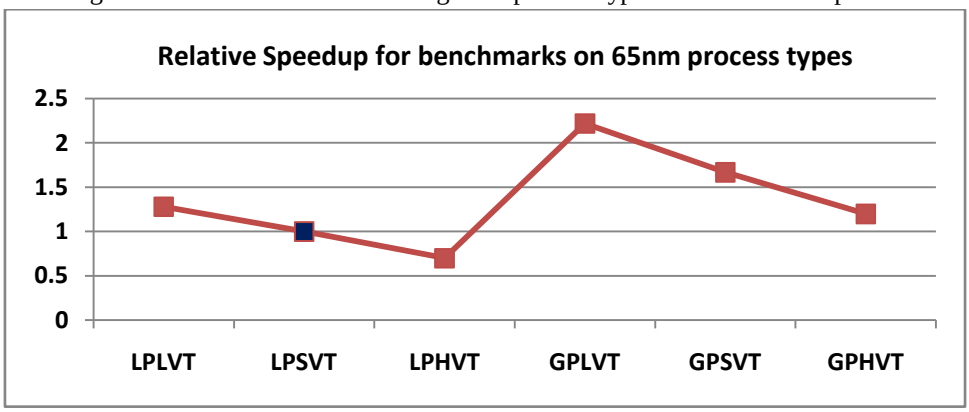


Fig. 4.35: Relative speedup for benchmarks for different ST65nm process node types

4.2.5 Miscellaneous Experiments Perspectives

This section provides some miscellaneous experiments frontiers, for which the basic study and infrastructure was almost completed in the thesis work but no exhaustive experiments were conducted (due to timing constraints and heterogeneity of contributions) as were elaborated throughout the chapter. This section briefly highlights them and will also facilitate discussions of chapter 6 for future outlines.

1- Feedbacks optimization

Throughout our explorations of SB-eLB multiplexers, the feedbacks of the logic block (eLB) were kept fully connected (SB-eLB <=4U+**eLBout**). While for small cluster sizes the value of eLBout compared to channel size (U = W/2) is often very small, however as cluster size increases this value starts getting bigger. Furthermore in any case, the fully connected structure connects each LUT (BLE) output to each and every input of all the LUTs in the cluster. This leads to over flexibility that is panelizing in terms of area (and also speed, based on mux size change). In section 4.1.3 we analyzed details of feedbacks statistics (figure 4.9) which clearly indicated the penalizing nature of fully connected feedbacks. In a similar fashion like SB-R and SB-eLB explorations, the investigation of feedbacks depopulation while maintaining good routing efficiency will be interesting and are not extensively addressed in research literature. Appendix A1 provides some examples of state of the art for inspiration in this regard.

2- SB Topology (Diversity) and Flexibility

Throughout our experiments in this thesis work we used disjoint SB which is most widely used/published topology among FPGA researchers. The SB topology itself is also a wide research area and several kinds have been proposed, [Wiliton97, 4.25][Lemieux&Lewis02, 4.21][Fan, Cheung et al03, 4.92][Fan, Cheung et al04, 4.93] provide a good general overview of efforts done. In our SB-R and SB-eLB explorations we observed how diversity helped achieve superior results; it will be interesting to conduct some new experiments utilizing the eFPGA Creator customization infrastructure to investigate benefits of diversity which was a major motivation for above mentioned works. Furthermore all the experiments that we conducted were on a square eFPGA, in reality eFPGA will/may often have rectangular shape in SoC. It will be essential to investigate challenges for irregular SB (the horizontal and vertical channels of different size). [Altera_Lewis,Betz et al03, 1.26] highlights the issues of directional bias for rectangular FPGA cores, where it is more silicon efficient to have horizontal and vertical routing channels of different sizes.

Throughout our experiments we used the Fs = 3 for SB flexibility. This value has almost remained de-facto in published works as it provides ease of pattern (topology) design and good-enough routing flexibility. It will be interesting to investigate different values of Fs (homogeneous, heterogeneous in SB) in connection with the diversity of SB topology to explore different what-if cases.

3- Effect of LUT and Cluster size

In the presented explorations the work was strongly focused on LUT size 4, cluster size 4 (based on motivation; fix the basics at some value and innovate routing for it). Several innovations and attractive results were obtained from the explorations. It will be interesting to extend these explorations learning (diversity etc.) into generalized algorithmic forms to perform a wider design space exploration for different LUT and cluster sizes to achieve further optimal architectures. A basic revisit to investigations of section 4.1.2 (LUT size) for the fundamental architecture style with timing was performed (see appendix A2), which revealed *for our current architecture style in raw form LUT3 and LUT6 are inefficient compared to LUT4, while LUT5 showed best overall properties*. It will be interesting to further investigate that with optimal (golden tiles-like) patterns for a wider range.

4- Explorations with Architectural Heterogeneity

Last but not the least, a significant frontier of FPGAs to combat the silicon gap is architectural heterogeneity, which is common in all commercial FPGAs, but is still in very early stages in academic research. Several efforts are on way but breakthrough of the magnitude of VPR of late 1990s has still not happened in this regard and the research gap for heterogeneous FPGAs (multitude of complex hard blocks, carry chains, multi clock etc.) between commercial FPGAs and academic research is continuously increasing. [FPL09Keynote_Rose, 4.4] has also addressed similar rising challenges issues. It will be one of the biggest and interesting research challenges for eFPGA to provide the complete motivation theme of customization for target. It will be interesting to perform first stage experiments in following dimensions.

Long Wires: Are well addressed in research work for VPR-like architectures. New experiments for eFPGA architecture in connection with SB explorations (topology, flexibility etc.) using the knowledge of section 4.1.4 and facilities of eFPGA Creator will be interesting.

Hard Macro Blocks: Performing first level experimentation for the heterogeneous building blocks of eFPGA (chapter 3) in relation with long wires and SB explorations (routing challenges of hard blocks) will be obligatory. The hard blocks exploration can be divided into two types/classes; general purpose (columns of basic blocks of memory etc.) and application specific (larger blocks, primarily on the boundary of core).

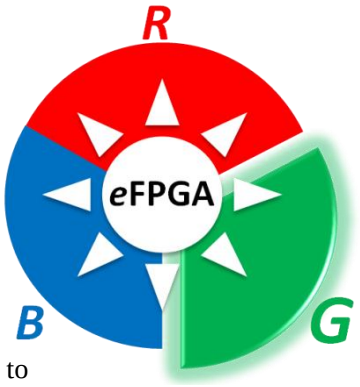
4.3 Summary

This chapter presented in detail the contribution B (chapter 1) of the thesis work related to architectural exploration of eFPGAs to create efficient customized architectures. It explained in detail the fundamental explorations in section 4.1 related to the effects of LUT size (along with mapping efficiency issues), cluster size, channel size challenges, routing analysis, the effects and challenges of power consumption in beyond 90nm technologies and some basic comparisons with state of art for inspirations. Section 4.2 extensively addressed second level of customizations to create more silicon efficient SB with negligible tradeoffs in routing efficiency. The work was divided into two main parts of SB; the SB-R and SB-eLB respectively. With successive experiments based on knowledge of chapter 2, section 4.1 and facilities of eFPGA Creator and eFPGA Programmer, architectures of finer quality with attractive properties were explored and finally joined together for obtaining best of both. A comprehensive overview of partially ongoing and near future experiments was also presented. The key points observed/discussed can be summarized as follows.

- The eFPGA architecture with unified SB (chapter 3.1) was explored in detail on the MCNC benchmarks with CAD and silicon explorations. Several tools were made to facilitate explorations
- The mapping efficiency decreases with increase in LUT size (the phenomena is quite universal)
- The area of LUT is negligible compared to interconnect (routing) area and routing demands do not grow in same proportion like LUT size. Motivation of bigger LUT sizes
- For the explored eFPGA architecture style in *raw form*, LUT4 is more silicon efficient than LUT3 and LUT6 (for current mapping efficiency of eFPGA Programmer), LUT5 provided attractive results
- The channel size demands do not grow in same proportion as increase in cluster size (motivation for larger clusters), but lack of an intelligent intra and inter cluster routing architecture can nullify exploitation of this potential
- The channel size demands highly fluctuate among benchmarks, making it most complex architectural parameter to manage for a range of benchmarks while having an optimal architecture
- Power consumption is a huge challenge for beyond 90nm nodes. The power vs speed tradeoff is significant at these nodes. The voltage scaling is flattening and static power is continuously getting higher. The threshold voltage can provide some benefits for static power at cost of speed
- The unified SB based architecture with eliminated conventional fully connected local interconnect in raw form is inefficient compared to VPR-like CB+LI based architecture. By making an effective use of diversity of eLB (logic block) inputs and outputs connection to the routing network in SB can significantly improve architectural efficiency which is competitive to the classical widely explored VPR-like architectures
- The presented explorations provided some key observations that can help in exhaustive revisit for these and several other possibilities (including more efficient two layer SB-eLB structure, eLB outputs logical equivalence investigations by mux at eLB out etc.) and other sate of art architecture styles
- The explorations provided efficient pure soft eFPGA architectures with reasonable parameters reaching logic densities of the range 400+ LUT4/mm^2 on 65nm
- FPGAs/eFPGAs are masterpiece of transistor waste (like microprocessor+ram), even the finest tuned architecture is/can be highly inefficient. The potentials of FPGAs (indirectly eFPGAs) are very high and promising (chapter 2), in addition to the classical, beyond classics techniques must also be investigated by researchers to further improve different architectural challenges

The next chapter presents the eFPGAs in systems, experimentation of eFPGAs as reconfigurable accelerators, system integration challenges and an interesting case study for emerging MRAMs perspectives for eFPGAs. The results presented in next chapter [S-7] are based on section 4.1 [S-9].

Chapter 5: eFPGA in Systems [G]



Since the thesis focused on embedded FPGAs, it was essential to have basic experiments for system integration in addition to eFPGAs architectural exploration (chapter 4), to have the idea of system integration challenges and potentials of eFPGAs in systems. This chapter will address general motivations and potentials of eFPGAs for SoCs, potentials for reconfigurable acceleration with eFPGA with its standard RTL programming flow and benefits of using ESL tools due to that. Then greater part of the chapter discusses HW/SW co-design experiments done by using eFPGA as reconfigurable accelerator with two processors (Plasma and LEON3) [S-7][S-8]. The work with LEON3 processor (www.gaisler.com) will be discussed in detail. Two major questions arise while using a reconfigurable accelerator, programming complexity and silicon tradeoffs. The work has tried to address both issues by investigating ESL for added programming ease and complete silicon tradeoffs analysis for area, power, and speed on ST65nm low power (LP) process. The chapter also presents an interesting case study of the perspectives of beyond classics MRAMs based eFPGAs for SoCs [S-4][S-5]. Finally it presents briefly the test chips at 130nm (MRAM configuration), 65nm (Latch/SRAM configuration) that were made based on eFPGA architecture (chapter 4.1) of this thesis work. The test chips themselves are not direct contribution of this thesis.

5.1 General Motivations

This section discusses potentials enabled by the eFPGAs in SoCs and a revisit to the potentials of eFPGAs as reconfigurable accelerators which can be a prominent application of eFPGAs in SoCs.

5.1.1 eFPGAs potentials in SoCs

The fundamental motivations of eFPGAs are well known and potentials are undeniable. We saw in detail in chapter 2 that there have been several attempts in industry in this regard since late 1990s (both from startups and leading FPGA vendors) and tried to address that some things have changed over time compared to past and eFPGAs can now have a strong potential in industry. Below are some well known and some added benefits observed during research of this thesis work regarding eFPGAs.

1- Product Differentiation, Time to market

The most prominent benefit for which historically (even now) eFPGAs were proposed or found potentials in industry was issues like product differentiation and fast time to market. The increasing complexity of SoCs and their huge development costs at beyond 90nm nodes is making difficult to design dedicated SoCs for each and every application. They require some post manufacturing flexibility. This allows amortizing high design costs over several end applications. The potentials of FPGAs in such regards are well known. eFPGAs bring these benefits right inside the SoCs providing a flexible hardware capability in addition to flexibility by software which is very common among the SoCs through a processor [S-1]. That also allows SoC vendors to rapidly create differentiated solutions among their own products and catching up with the competitors by bringing their solutions fast to the market.

2- SoC prototyping (testing new ideas)

FPGAs are de-facto prototype devices due to their fine grain nature. Large emulation systems (multiple FPGAs) or stand alone very large FPGAs (with hundreds of thousands of logic blocks and hard blocks) are in wide use for prototyping SoCs. As we discussed in chapter 2, with the advancements of architectures and continued increase in logic density due to Moore's law, FPGAs have become programmable platforms and are directly used as a programmable SoC for many applications where their tradeoffs in terms of cost,

power and speed are acceptable compared to high costs of designing a SoC. If we thinking in an opposite direction (FPGA vs eFPGA scenario), an eFPGA can be beneficial for prototyping SoCs. SoCs usually are made up of in-house or 3rd party IPs interconnected (e.g. AMBA etc.) with each other and controlled by a processor (e.g. ARM, MIPS etc.). Most of the IPs or designs remain same when moving to next node or making new experiments with some new differentiated IPs. Having some eFPGAs in the SoC can help in this regard, making SoC itself its prototype platform [S-2]. eFPGAs can/may be removed in the final mass production depending on requirements, or can be kept or reduced in size/specifications to provide the post manufacturing benefits of differentiation and time to market discussed above.

3- Multiple possibilities (differentiation, experimentation, reconf. acceleration etc.)

With FPGA-like fine grain architectural nature, multiple eFPGAs of different sizes and architectural specifications can be used in a SoC to provide multiple benefits. Figure 5.1 shows a theoretical concept diagram showing SoC with multiple eFPGAs. From the figure we can also observe the discussions of above points and see opposite nature of eFPGA compared to FPGA. In most cases of FPGA designs, huge portion of expensive fine grain resources are used to implement fixed IPs from 3rd party. Only few differentiated IPs are written for which the flexible nature of FPGA provides an ease of design and integration. We saw in chapter 2 that newest trends of FPGA vendors of putting hard processor blocks and ever increasing hard macro blocks also illustrates the IP based design nature of SoCs where most of the design is fixed and having small amount of fine grained flexibility is a better proposition from cost, performance and power point of view. Figure 5.1 shows three different case scenarios for eFPGAs in SoC which complement the above discussions. A stand alone eFPGA Core of suitable size (top right) is good for product differentiation. A larger eFPGA Core (bottom) can provide several benefits like product differentiation, experimenting new ideas, I/O interfacing and much more. Furthermore due to parallel nature of eFPGA hardware (like FPGAs) several things can be done in parallel and independent of each other, a potential harder to obtain from other non-FPGA-like programmable solutions. Finally the reconfigurable acceleration; although in principle other cases are also performing the same thing (processor being the master) but some specific eFPGA with architecture tuned to target application domains can be very attractive for co-processing the intensive computations of processor in parallel to provide performance enhancements and power reduction which is already becoming mainstream in high-end computing for servers with processor + FPGA computing, eFPGAs can bring a similar benefit on a smaller scale in embedded systems SoCs (*we will discuss this type of experiments in detail in this chapter*).

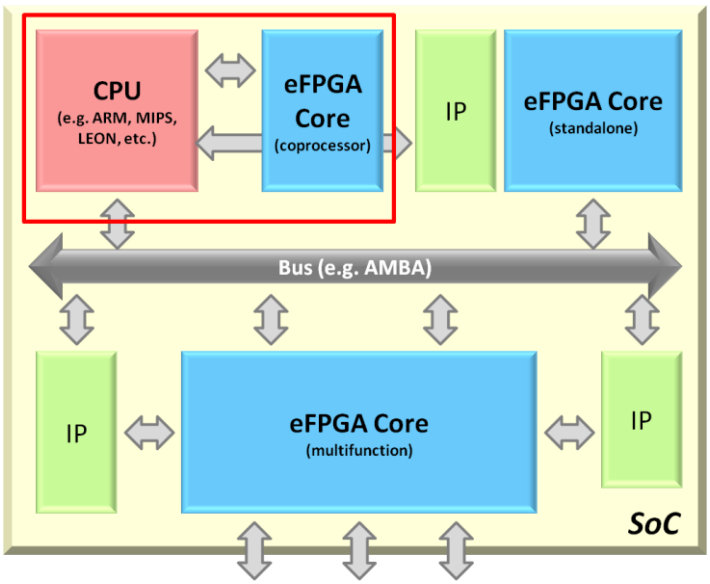


Fig. 5.1: Concept SoC scenario with eFPGAs

4- Soft eFPGAs (technology independence)

As eFPGA is pure soft it can be ported to any fab and process node, so the benefits of eFPGA can be immediately obtained without the bottleneck of a fab-specific or node-specific hard eFPGA solution. Since eFPGAs are supposed to be of small size and are highly customizable and technology independent, such features in many cases can provide a good silicon tradeoff between hard eFPGA and soft eFPGA. With additional possibility of creating a more efficient semicustom or full custom solution of eFPGA (hard eFPGA) based on market size of end product. Hence providing dual benefit of immediate availability of soft eFPGA and migrating it to hard if market size is large and justify ROI (Return On Investment).

5.1.2 eFPGAs as reconfigurable accelerators

This section provides a brief overview of potentials of reconfigurable acceleration using eFPGAs in general and in particular in the light of eFPGA RTL programming flow, customization capabilities and new trends and challenges in industry.

1- Not new concept, mostly failed, why again

We addressed in chapter 2 in detail about the pros and cons of different programmable technologies and why many of them failed while FPGAs survived and continued to become stronger and stronger. In the area of reconfigurable acceleration huge work is done in industry and academics (particularly in academics). Several novel and innovative solutions were proposed, but is well known that most of them failed or never found wide acceptance in industry [Brunelli08, 4.89][Hartenstein01, 4.64][S-3][S-7]. From motivational point of view, that historical perspective does not provide an enthusiastic start for a researcher and particularly for a company (case for this thesis work). To try to find answers to what might be the reasons of failure is more important than jumping into the same adventure again, going in same direction (there was nothing wrong technically with most of past efforts), doing the same thing, coming up with a similar solution like past. And finally finding a nice place in the famous graveyard of Reconfigurable Computing and IEEE/ACM/EEtimes memorial of citations/news which after few years of death is the only source remaining for new researchers to study those efforts. As investigating technologies and digging deep into failure reasons is one of the three foundation axes of this thesis work (chapter 1), so while conducting basic research experiments [S-7][S-8] in this area, efforts were made to investigate in industrial scenario what might be the obvious and probable reasons of failures in this direction in past. The thesis work has tried to visualize the new waves of challenges and trends in industry and investigating what opportunities they bring to enhance potentials of solutions in this area (chapter 2). Next sections along with scientific experiments will also try to investigate some of these issues.

2- Standard RTL flow and rise of ESL tools

We saw in chapter 3 that the programming flow of eFPGA is standard HDL based like is in commercial FPGAs. This gives high ease to programmer for programming eFPGAs, plus it also allows re-using of several existing HDL (VHDL/Verilog) functions, IPs to quickly implement them on eFPGAs (the IP re-use benefits of FPGAs which we discussed in Chapter 2). Furthermore as the flow is RTL based eFPGAs automatically can enjoy the benefits of ESL tools which further help the programmers to write their applications at a higher level in ANSI C/C++ (also SystemC) and ESL tools automatically transforms them into RTL comparable (in some cases even better, in design space exploration scenario) to optimal hand coded HDL.

This gives *dual benefit of RTL programming model* of eFPGAs. We discussed in chapter 2.3 that programming model was one of the most prominent reasons of failures of several solutions in past, where due to lack of standards and obvious design mapping complexity, the solutions had to create a new language or a flavored version of a standard language to allow their compilers to optimally map applications to those hardware. RTL flow of eFPGA allows taking benefit of mature state of the art synthesis tools and now as industry is highly focused on benefits of ESL, this additionally gives benefit to

take leverage of commercial ESL tools which essentially deal standard RTL flow, giving the programming ease benefit which was a main motivation of several past works.

3- Out of Box thinking: use weakness as strength

FPGAs are well known for high power consumption. Apparently it might really sound very strange to think about FPGAs to help reduce power consumption. However FPGAs have remarkable other properties which have only been partially used or exploited in industry. We addressed semiconductor industry markets in chapter 2 and saw that FPGAs are still a small niche (nearly 2%) of semiconductor industry. Why and where FPGAs are used is well known (we saw some details in Chapter 2) but if we analyze how they are used, it becomes quite evident that most of the time they are implementing static functions, which sometimes can be upgraded which makes FPGAs attractive but rarely or in very limited cases they are actually used as a reconfigurable platform, which also explains the products like Structured ASICs (FPGA to ASIC migration) which are provided by leading FPGA vendors also. FPGAs (although there are other technologies like coarse grain etc.) represent the most prominent symbol of reconfigurable devices but it is true that rarely they are ever used in industry to perform Reconfigurable Computing. On the commercial side Xilinx was the only FPGA vendor that provided features like dynamic reconfiguration (Now Altera too is providing in 28nm Stratix V) which clearly indicated that there was not a high demand in industry for such features and Xilinx also acknowledge not wide spread use of dynamic reconfiguration [Xilinx07_Trimberger, 2.6].

This second power of FPGAs to be really used as a reconfigurable computing device holds prominent potentials to address some key problems in industry (in academics it is a large research area and hundreds of interesting papers in conferences can be found, all based or limited to features provided by Xilinx FPGAs due to obvious reasons). As our work is focused on eFPGAs, this gives additional opportunities to exploit such features with customized eFPGA IPs, which exploit benefits of reconfigurable accelerators. In next sections we will show our work of using eFPGAs as reconfigurable accelerators using conventional architecture to explore potentials of eFPGAs to enhance performance and save power which is becoming very crucial in industry. At the end of the chapter we will briefly discuss how some future technologies (MRAMs) can provide significant opportunities for exploiting reconfigurable accelerators like non-volatility, dynamic reconfiguration and more importantly multi-context on classical CMOS process with very low silicon overhead compared to conventional architecture.

4- Basic Experiments done with Two Processors (Plasma & LEON3)

Prior Art: There are different research works done in the past in academic research regarding potentials of reconfigurable accelerators which provided some motivations and inspirations for these experiments. [Brunelli08, 4.89] provides a survey of some of the approaches like XiRisc [4.104], Garph [4.105], PipeRench[4.106] etc. Chapter 2.2 provided some details and issues for coarse-grained-like architectures.

eFPGA Differentiation: What distinguishes this thesis work from several past research efforts is that eFPGA is conventional FPGA-like architecture, which in general is useful for multiple purposes (reconfigurable acceleration is one of them but not the only one!). It is useful for wide range of applications for SoCs, utilizes standard programming flows (RTL). With further distinction of potentials of ESL (due to RTL flow) exploitation for addressing programming complexity issue (was among major motivations of compilation based reconfigurable accelerator efforts in several past solutions).

An exhaustive reconfigurable computing experimentation is beyond the scope of this thesis work. These experiments were performed in the contribution G (chapter 1) aspect of this thesis work to investigate how sound eFPGAs might be from some system scenario stand point in terms of silicon tradeoffs, programmability etc. The simplest, straightforward and yet very important aspect to investigate was reconfigurable acceleration, which brings the well known and addressed HW/SW co design scenario. We conducted the experiments keeping the motivations of past works in mind and tried to address the

challenges on the base of survey knowledge of this thesis work (as explained above). The experiments were conducted with two processors, a free MIPS-like Plasma/Mlite processor from OpenCores (opencores.org) and LEON3 processor, presented in [S-7][S-8]. Only the work with LEON3 [S-7] is described in detail in next sections as it was more detailed and superset of the work with Plasma [S-8].

5.2 eFPGA with LEON3 Processor

This section describes in detail the work of eFPGA integration with LEON3 processor as a functional unit and a co-processor unit for experimenting use of eFPGAs as reconfigurable accelerator for performance enhancement and power reduction (with analyzing silicon tradeoffs) along with the ESL to investigate the programming ease potentials [S-7]. The eFPGA used in this work (LUT size 6, cluster size 4, channel size 32) was based on work of [S-9] which was discussed in chapter 4 (section 4.1).

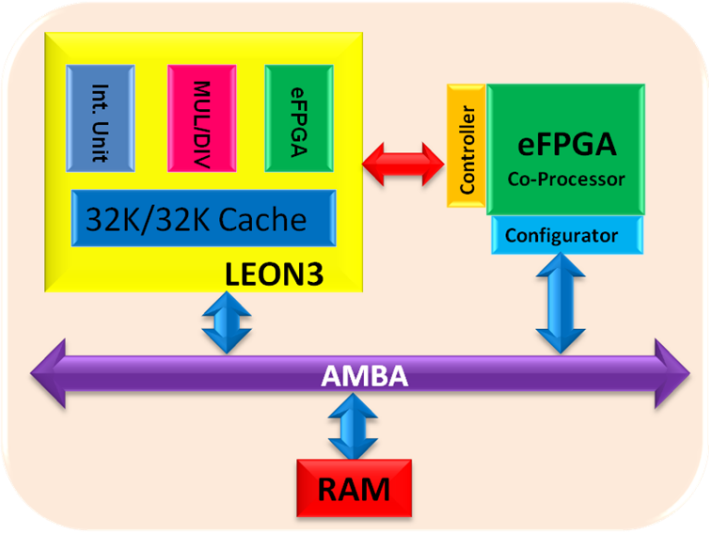


Fig. 5.2: eFPGA integration with LEON3 as functional unit and co-processor

5.2.1 eFPGA-LEON3 System Integration

The LEON3 [5.22] is a configurable processor core written in VHDL and uses the SPARC (Scalable Processor ARChitecture) instruction set (SPARC V8 manual) [5.23]. The advantage of the availability of its source helps to make modifications to explore new concepts. eFPGA was integrated with LEON3 processor in two ways; as a functional unit and a co-processor unit as shown in figure 5.2. Brief details are described below.

1- As a Functional Unit (processor pipeline)

Figure 5.2 shows eFPGA integrated inside the LEON3 processor as a functional unit. We integrated it like the Multiply/Divide unit. We created a new custom instruction for eFPGA which allows executing data in eFPGA creating a virtually reconfigurable instruction whose execution is based on the programmed hardware inside eFPGA.

2- As a Co-Processor Unit

On the right hand side of figure 5.2 eFPGA integration as a co-processor unit is shown. In LEON3 the co-processor interface is only partially implemented. We created the entire interface based on the SPARC manual on which the LEON3 processor is based and designed. According to the manual the co-processor interface is similar to FPU (Floating Point Unit) interface but is flexible and custom dependent. We only created subset of interface which was needed by us (simple data transfer to/from eFPGA). This also helped avoiding unnecessary silicon overhead and required very few cycles to send and receive data. We only implemented 8 registers in the register bank and only used basic Load, Store and CPOP (co-processor operate) instructions of SPARC V8 manual [5.23].

This setup created a basic Processor + Reconfigurable accelerator platform to perform different experiments and also to analyze the pros and cons of functional unit vs co-processor unit in technical and commercial perspective. The eFPGAs are configured using a compact configuration unit which loads the configuration bitstream from main memory into the eFPGA through the AMBA bus at the start of execution. Figure 5.2 shows a configuration unit attached to the co-processor eFPGA. Similar unit is attached with functional unit eFPGA. The configuration units are controlled by LEON3 processor through software providing ease of use and flexibility. The configuration portion and configuration load time is not the part of this discussion, as this work did not perform run time dynamic reconfiguration but independently analyzed several possibilities step by step as we will see in section 5.2.3. The eFPGA was loaded with the configurable hardware in the beginning of execution and then comparisons were made on execution times as we will see in details in sections below.

5.2.2 Experimentation Flow

The experimentation flow is shown in figure 5.3. It starts with an application written in ANSI C/C++. The application is profiled for identifying the computation intensive critical areas of the application. The Hardware-Software (HW/SW) partitioning is done based on the profiling knowledge. The extracted critical computation is transformed into HDL by using ESL tools (and/or manual HDL coding). The software is compiled and run on LEON3 and hardware is mapped on the eFPGA using the eFPGA CAD tools and finally the co SW + HW execution is done and the performance enhancements and silicon tradeoffs are observed. Brief details of the steps of exploration flow of figure 5.3 are described below.

1- Application Profiling (HW/SW partitioning)

The first main step of the flow is HW/SW partitioning. To facilitate this step (ideal case would be automatic) we created a simple profiling tool with the help of SimpleScalar modeling tool [5.25], which analyzes the application on a MIPS-like model. Although LEON3 is not a MIPS instruction set processor, but to get a profiling look at the execution of application it does not really matter too much and we will see in experiment results that the knowledge gained from the profiling tool was quite close to what we actually achieved on real execution. To further facilitate visualization of the profiling tool results, we created an HTML based user friendly output of results to easily visualize the output of profiler, like critical functions, level of their criticality, overview of every single C instruction of the application from computational criticality, execution trees etc. A snapshot of the outputs of the tool is shown in figure 5.4 for a feeling of user-friendliness of the tools. The details of the profiling tool are beyond scope of this discussion. The tool provided a great ease of analysis of applications (although being approximate) which would have been very time consuming and difficult if would have been manually done by simulating application on LEON3 with special arrangements to get a similar overview.

2- Application mapping (use of ESL)

The major concern while conducting HW/SW co-design like experiments is the programming of the reconfigurable unit. As the programming model of eFPGA is RTL based (chapter 3), it provided great help in this regard. We investigated this property in two ways. First the classical and straightforward way to write the HDL (VHDL in our case) of the partitioned hardware, implement it on eFPGA with the eFPGA Programmer (renamed, was Niagara in [S-7]). Second exploit and investigate the potentials of ESL tools to automatically transform the partitioned C/C++ code to RTL and map it to eFPGA, hence eliminating the manual HDL coding need. For this purpose we used Catapult ESL tool of Mentor Graphics (mentor.com).

3- Analyze Silicon tradeoffs (Area, Power, Speed)

Another major concern regarding use of reconfigurable accelerators after programming issues is silicon tradeoffs. It is pretty straight forward that processing in parallel yields to higher processing throughputs, so observing and presenting just speedups achieved (often is/was the only thing addressed in such works) with using a reconfigurable accelerator is not a good and fair contribution, in fact it might even be

pointless in realistic scenarios. To try to fully visualize the scenario as much as possible, we investigated silicon tradeoffs in terms of area, power and speed on a 65nm low power process of ST. That gives a more clear idea of the criticality and challenges regarding use of reconfigurable accelerators.

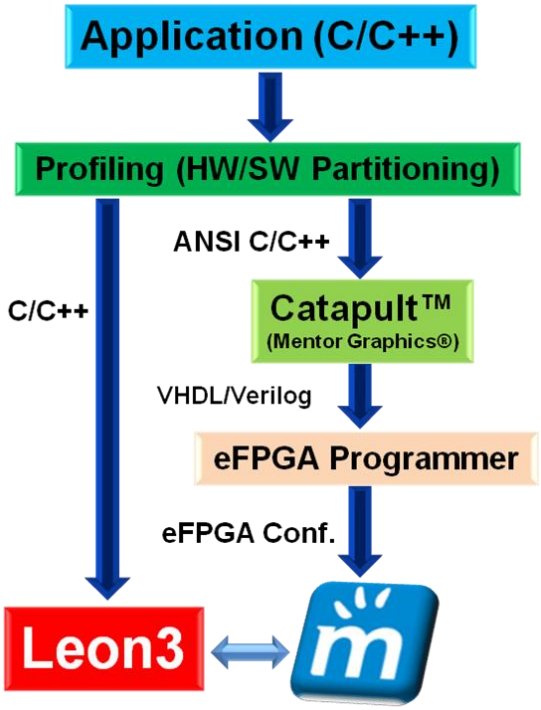


Fig. 5.3: Experimentation flow

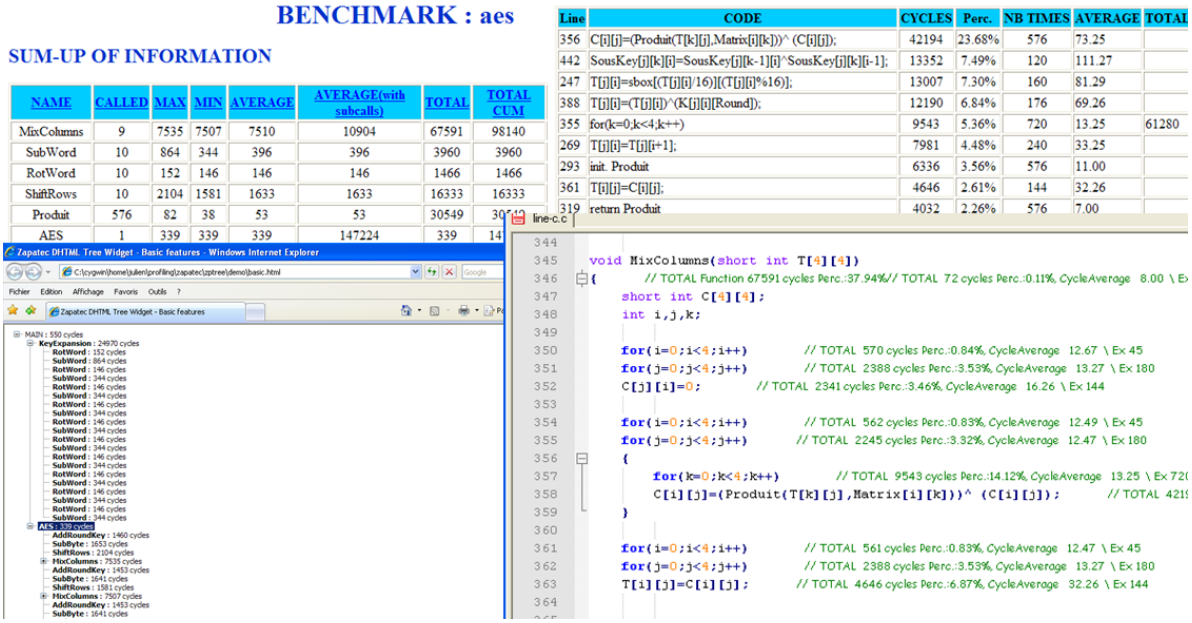


Fig. 5.4: Profiling AES Application Critical function

5.2.3 Experiment with AES algorithm

This section presents the details of experiments done with AES (Advanced Encryption Standard) application using the flow discussed in previous section.

1- Profiling AES

In the first step we profiled the AES application with our profiling tool discussed above and we will see below how advantageous this tool was for making decisions and conducting experiments. Table 5.1 and Table 5.2 show the main results of the profiling for the critical functions and critical instructions of the application.

Table 5.1 indicates that MixColumns is a critical function in AES application accounting for almost half of the computation load of the application. Table 5.2 shows the top 4 most computation intensive instructions in the application. It show that line 356 of the code is most computational expensive accounting for almost 25% of the execution time. Figure 5.5 shows the code of the MixColumns function which was identified as the critical function from Table 5.1. We see that the instruction identified as most expensive from table 5.2 is at the heart of the function in deep nested loops. To investigate the benefits of reconfigurable acceleration keeping silicon tradeoffs in perspective we divided this function into 4 Steps as shown in figure 5.5 and step by step implemented and investigated them for performance gains and silicon tradeoffs. Next sections explain the details.

2- Mapping HW/SW partitioned portions on LEON3+eFPGA

We implemented the critical function shown in figure 5.5 in hardware in four gradual steps to see the overall gain in terms of area, power and speed. These four steps are as follows. In step 1 we implemented only Product function in hardware. Table 5.3 and 5.4 show the results of this implementation for number of clock cycles if eFPGA in pipeline is used vs if eFPGA as a co-processor is used along with the eFPGA hardware resources needed (which are off course same for both cases). The AES algorithm takes 40358 clock cycles for pure software execution on LEON3 processor, by moving Product function to eFPGA, number of required clock cycles came down to 30430 for pipeline implementation and 30447 for co-processor and gave a speedup of almost 1.3X. The eFPGA took only 41 LUT6 for hand coded VHDL and 44 LUT6 for Catapult generated VHDL. In step 2 we implemented the "xor" in addition to Product function, in step 3 we implemented the inner most "for" loop and finally in step 4 we implemented the complete function. Results for all these implementations are shown in table 5.3 and 5.4 with the speedup achieved in all these steps and the used hardware resources of eFPGA for both hand coded VHDL and Catapult generated VHDL. It can be observed how the gain in performance gradually increases by transferring more and more computation to eFPGA. However it is very important to note the relative increase in hardware resources. We can see that step 3 is giving a good speedup with very small amount of eFPGA resources (only 67 LUT6 for hand coded VHDL).

Another interesting observation in table 5.3 is that the speed-up achieved with integrating eFPGA inside the processor pipeline and using it as a co-processor is almost same on overall application level analysis. Individually for execution there is off course a difference because with co-processor interface we have to spend some additional cycles to load data into co-processor registers and there is some further delay which is caused by co-processor controller state machines. For our case the difference is much less also because we created a very compact and fast co-processor interface custom to our needs which requires very few clock cycles for performing the data transactions between processor and eFPGA. For a more realistic and standard case the latency of data transfers will increase but is interesting to note that it will still be more attractive and efficient solution compared to an eFPGA accelerators connected through the Bus (figure 5.1) which have their own particular advantages with relatively larger eFPGAs but will be much less efficient for tightly coupled accelerator with the processor (due to the obvious scenario with Buses). Furthermore it

also gives some insights of benefits of co-processor vs functional unit approach from technological and commercial perspectives. We will discuss it in more detail in section 5.3.

Frequency issue: It is important to note that for these experiments it was assumed that both LEON3 and eFPGA are running at same frequency (100MHz, as we will see in next section). In reality this might not always be the case (processors are usually much faster). However for our cases 100MHz for eFPGA was a good approximate assumption based on knowledge of chapter 4 experiments. As stated before these experiments were based on [S-9] eFPGA with limited knowledge about timing, with improved experiments of section 4.2 that we discussed, it validates that we had a good assumption about 100MHz eFPGA speed (we saw several much larger benchmarks running in around 50-100MHz range).

Function name	Called	Avg. Cycles	Total Cycles	% of execution
main	1	179838	179838	100.00%
AES	1	147224	147224	81.90%
MixColumns	9	10904	98140	54.60%
Product	576	53	30549	17.00%
KeyExpansion	1	30396	30396	16.90%
SubByte	10	1642	16422	9.10%
ShiftRows	10	1633	16333	9.10%
AddRoundKey	11	1453	15990	8.90%
SubWord	10	396	3960	2.20%
RotWord	10	146	1466	0.80%

Table 5.1: Profiling of AES application

Line	CODE	CYCLES	%	No. times Executed	Avg. Cycles
356	C[i][j]=(Product(T[k][j],Matrix[i][k]))^ (C[i][j]);	42194	23.68%	576	73.25
442	SousKey[j][k][i]=SousKey[j][k1][i]^SousKey[j][k][i-1];	13352	7.49%	120	111.27
247	T[j][i]=sbox[(T[j][i]/16)][(T[j][i]%16);	13007	7.30%	160	81.29
388	T[j][i]=(T[j][i])^(K[j][i][Round]);	12190	6.84%	176	69.26

Table 5.2: Profiling of all C instructions in AES Code

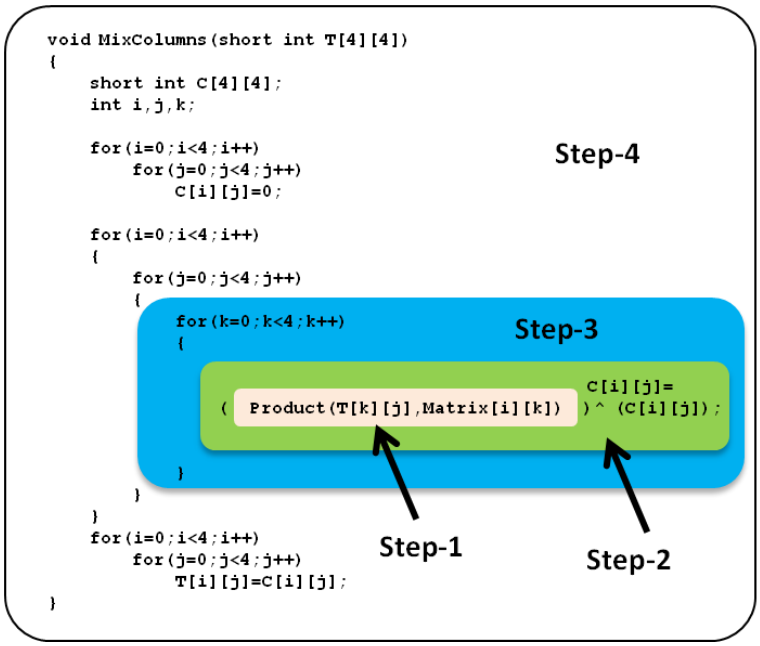


Fig. 5.5: Profiling AES Application Critical function

	With Functional Unit eFPGA (Cycles)	Gain X Times	With Co-Processor eFPGA (Cycles)	Gain X times
STEP-1	30430	1.326	30447	1.325
STEP-2	30722	1.313	30735	1.313
STEP-3	23752	1.699	23767	1.698
STEP-4	16244	2.484	16265	2.481

Table 5.3: Different LEON3+eFPGA Implementation Steps

Performance gain with functional unit and co-processor unit eFPGAs (Pure Soft LEON3 40358 Cycles)

	STEP-1	STEP-2	STEP-3	STEP-4
VHDL: Hand	11 Tiles (41-LUT6, 0FF)	10 Tiles (39-LUT6, 0FF)	17 Tiles (67-LUT6, 8FF)	127 Tiles (506-LUT6, 324FF)
VHDL: Catapult	11 Tiles (44-LUT6, 0FF)	18 Tiles (72-LUT6, 33FF)	33 Tiles (130LUT6, 78FF)	205 Tiles (819-LUT6, 604FF)

Table 5.4: eFPGA hardware resources for each step with ESL & Manual VHDL

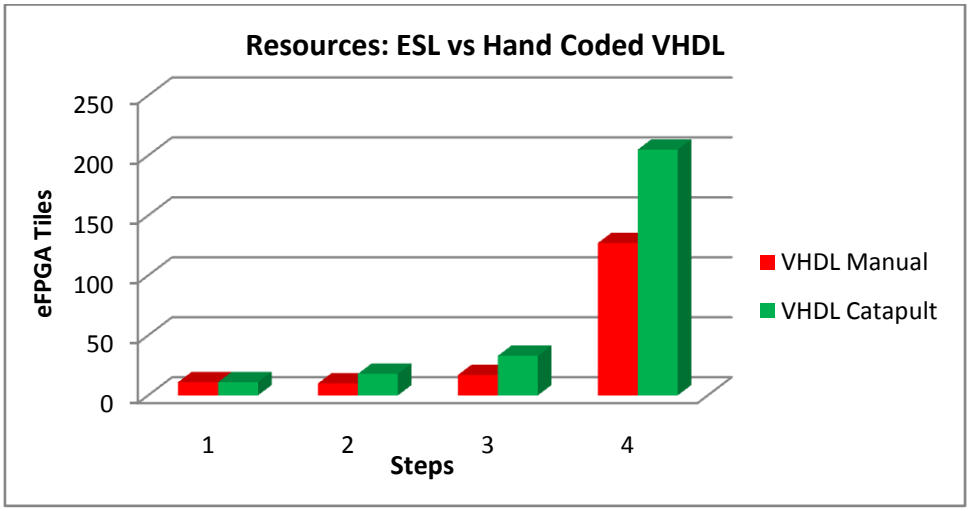


Fig. 5.6: eFPGA resources for ESL vs Hand Coded VHDL

3- Analyzing Silicon Tradeoffs at 65nm (Area, Power, Speed)

In this section we will analyze the silicon tradeoffs of the experiments. Table 5.5 presents synthesis results of LEON3 processor core (no FPU) on ST65nm LPLVT (Low Power Low Voltage Threshold) process libraries of ST Microelectronics provided by CMP [5.13]. For cache memory we used 32K instruction and 32K data cache (value usually found in processors of ARM and MIPS). For SRAM memory blocks for cache memory we used 65nm High Density (HD) low leakage memory blocks of STMicroelectronics provided by CMP. From the datasheet of memory blocks we found the static power (at 25^{o}C) and dynamic power (at 100MHz, with normal activity rate of 50%).

Analyzing the complete system for power with exact toggle activities was very complicated so as first step pessimistic power analysis approach was used (chapter 4) using the power estimation of tile through synthesis using Synopsys Design Vision at different toggle rates and static probabilities and multiply the results to number of tiles to get approximation of the core power. Through this pessimistic way the results are not accurate but give a good approximation value for having overview as in actual case power in most cases will be less than this pessimistic value as not all nets will be toggling, so if interesting results are observed in pessimistic case in reality they can be even better. Table 5.6 shows power statistics of an eFPGA core of 484LUT6 (based on [S-9]). It has LUT size 6, cluster size 4 and channel size 32 with logic density of approx. 320LUT6/mm^2, results are for 65nm ST process. Figure 5.7 shows a detailed overview of silicon tradeoffs of the experiment results of table 5.7. We see that step-3 for AES implementation

provides a good tradeoff. We also analyzed DES application; for it we achieved almost 10X speed-up by spending only 95 LUT6. From table 5.7 we see that we achieved it by just spending 0.296mm^2 of additional silicon due to eFPGA which only consumes approximately 4.18mW of total power. So the LEON3 processor which we found has maximum frequency of almost 185MHz at 65nm LP, with eFPGA it can be possible to get 10 times more DMIPS for DES in same frequency. If we apply Dynamic Frequency Scaling (DFS) while executing DES we can decrease the dynamic power of LEON3 almost 10 times by only spending around 4mW of additional power overhead of eFPGA. The very high gain in DES with very attractive silicon tradeoffs compared to AES is due to nature of the application. This also explains the challenges of having a high logic density for eFPGAs to provide a reasonable margin of coping with requirements of different applications.

Good Potentials: A very important point to consider here is that eFPGA at the moment has no power management and is completely soft core written in VHDL and is under continuous research to greatly enhance the architecture. Even at soft implementation level (which we are using for fast exploration and technology independence) and pessimistic power comparison, we can observe the promising advantages that can be achieved by adding a small eFPGA IP to the designs. Also we can observe in table 5.6 that our eFPGA has a very low static power (can be attributed to soft standard cell based nature of eFPGA with no pass-transistors etc. which usually are more leaky compared to full CMOS). Furthermore it is interesting to note that encouraging results are obtained even with basic customized eFPGA [S-9] (chapter 4.1) which was used in [S-7] on which these results are based (pre eFPGA Creator experiments!). We discussed several architectural enhancements in chapter 4.2 which led to more efficient and customized architectures. Revisiting these experiments using those new architectures will further enhance these results significantly in terms of area, power and performance. Furthermore section 5.4 will discuss additional beyond classic potentials to further enhance the capabilities of eFPGAs for reconfigurable acceleration.

65nm LP	Area (mm^2)	Static Power at 25°C (uW)	Dynamic Power at 100MHz (mW)
Core	0.191	85.3	5.75
32K/32K Cache	0.4	25.63	14.9
Total	0.591	110.93	20.65

Table 5.5: Area and Power consumption of LEON3 processor at 100MHz

65nm LP process: 1.0V, 25degrees, 100MHz	LVT	SVT	HVT
Static Power (mW)	1.27	0.105	0.011
Dynamic Power (mW) @ (Tr-0.25, Stp-0.25)	23	22	25
Dynamic Power (mW) @ (Tr-0.50, Stp-0.5)	47.6	46.6	53.8
Dynamic Power (mW) @ (Tr-1.0, Stp-0.50)	95.36	93.36	107.8

Table 5.6: Pessimistic Power Statistics of 484 LUT-6 eFPGA

	Speedup Gain X	Area (mm2)	Stat. Power 25°C (uW)	Dyn. Power 100MHz (mW)
STEP-1	1.3	0.128	8.61	1.804
STEP-2	1.3	0.122	8.2	1.716
STEP-3	1.7	0.209	14.07	2.948
STEP-4	2.48	1.578	106.2	22.264
DES	10	0.296	19.95	4.18

Table 5.7: Performance gains with Silicon Tradeoffs

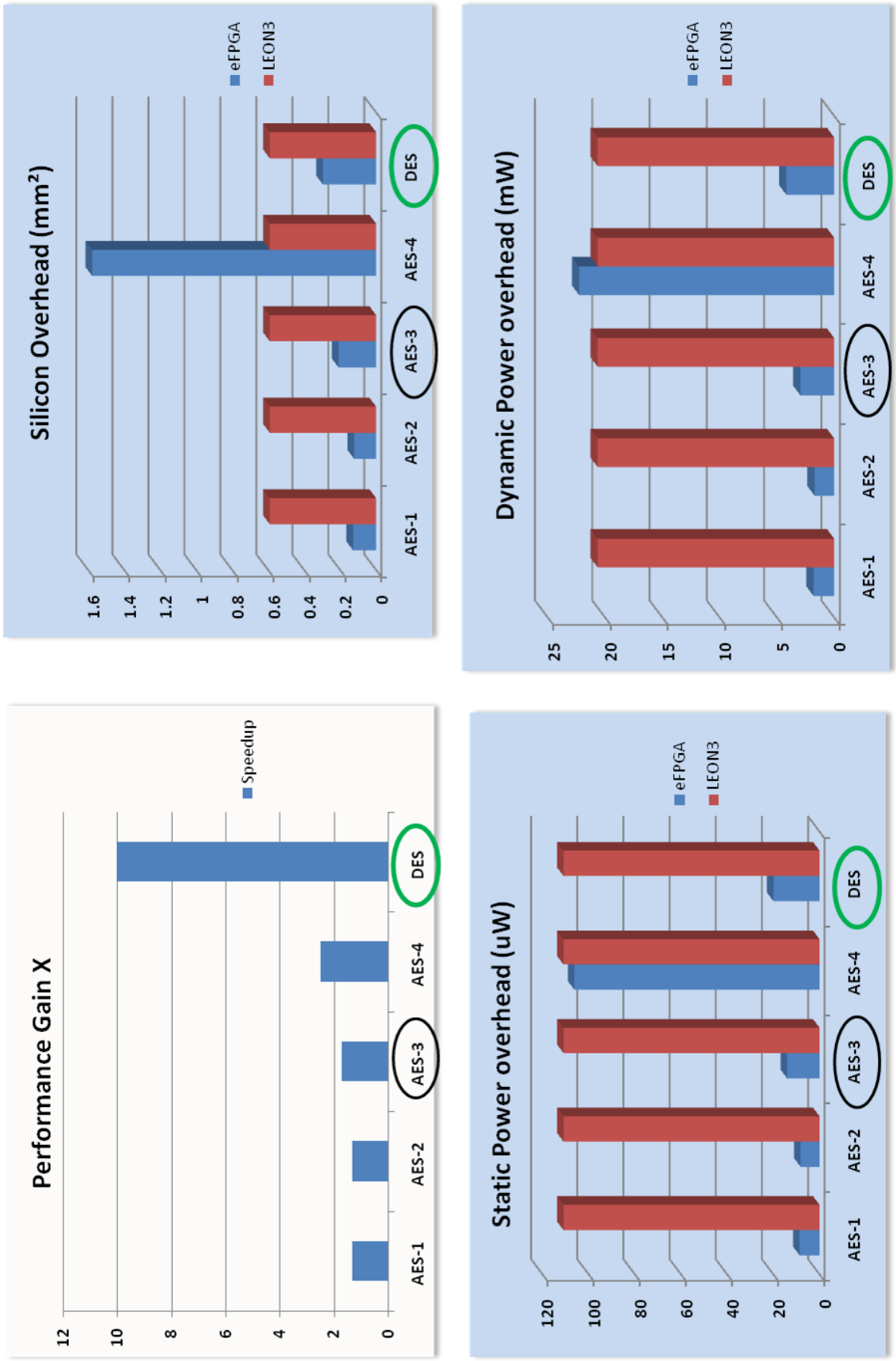


Fig. 5.7: LEON3+eFPGA silicon tradeoffs for different implementations

5.2.4 Benefits of ESL vs Optimal Hand coded HDL

In our experiments described above, we observed that the HDL generated by Catapult is giving close results to hand coded VHDL. It is widely known that with ESL the final RTL depends a lot on the way source C/C++ is written and the parameters of optimization given to the tool. We have found the same issues, it can be seen that in some cases the differences between hand-coded VHDL and ESL is larger than others (specially last 2 steps) because of the style of implementation. However we have found that programming through ESL is much faster and easier to verify. In case of Catapult the built-in support for different levels of verification and integrated support of ModelSim for simulation made it very easy to write the code and quickly verify compared to hand coded VHDL. Figure 5.8 shows some snapshots of Catapult tool, we can observe the integrated tools environments and design space exploration ease benefits. We can check several implementation options for target HDL at a higher level, like tradeoffs in area for decreasing latency and increasing throughput, pipelining etc. That is relatively difficult and time consuming to do at HDL level.

Furthermore we only had an access to an old version of Catapult from 2006. ESL tools have greatly improved over time, we observed in chapter 2 that ESL is a strong focus in industry at present, both the SoC designers and particularly FPGA designers/vendors have significant interest in the benefits of ESL; it might be interesting future investigation to revisit these experiments with the latest version of tools and cross comparing among different ESL tools (Synopsys, Mentor, AutoESL, Cadence etc.).

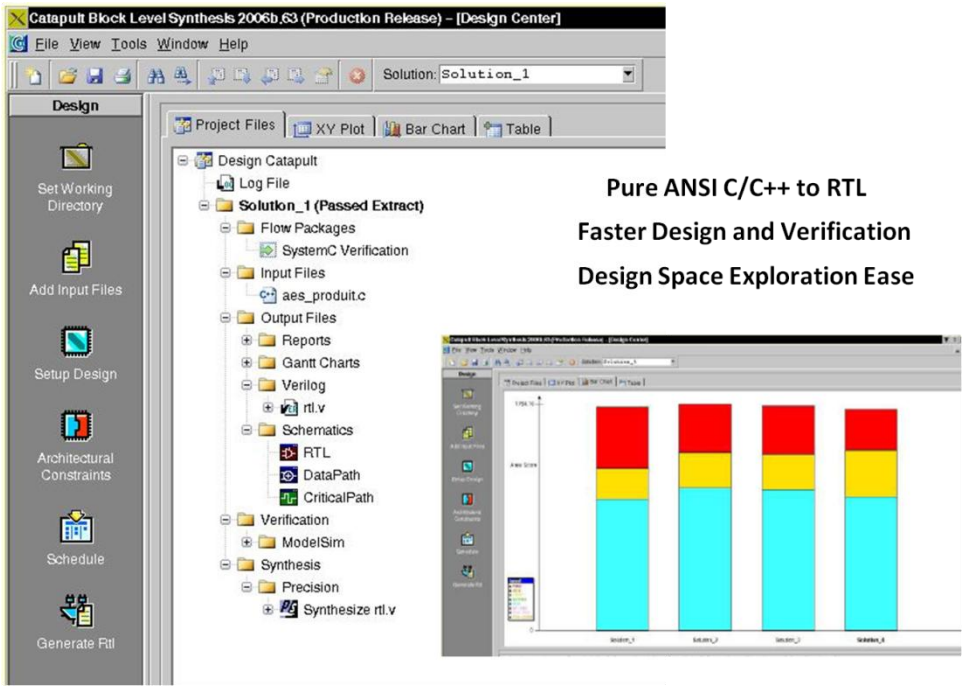


Fig. 5.8: Benefits of ESL compared to Hand coded HDL (Snapshots of Mentor's Catapult)

5.3 System Integration Challenges

This section briefly provides general motivations and observations learned from this thesis work regarding system integration challenges for eFPGAs in technical and commercial scenarios. First it provides general overview of the challenges for system integration. Second some basic efforts done to connect eFPGAs in system with AMBA buses are described. Finally it provides a view point for Processor inside FPGA vs FPGA inside Processor which has been often overlooked in research (particularly in academic), it tries to explain in technical and particularly commercial scenario why these two approaches apparently similar are often miles apart in reality.

5.3.1 General overview

This section highlights some of the general issues and challenges regarding eFPGAs system integration which were found/deployed/provide future motivations in the research of this thesis work survey investigations and practical experiments.

Importance of Standards: We observed in our discussions in chapter 2 that standards are very important in industry. Throughout the investigations and experimentations in this work this issue was kept in focus for motivations and research directions. We observed the importance of standard RTL flow and discussed eFPGA Programmer which provides a standard programming flow for the eFPGA. In HW/SW co-design experiments we investigated ESL use to address the issue of programming complexity of reconfigurable accelerator in a standard way. This has been a major focus of several RC (Reconfigurable Computing) researchers in past to explore an effective solution. We observed that dawn of ESL in industry is opening potentials to exploit several good points of past research efforts to create an effective HW/SW solution, with state of the art emerging tools support. Going one step further brings the challenge of how to integrate the eFPGA in SoCs in standard and effective way (as important and perhaps more critical than the software flows etc.). While investigating this issue several standard ways used in industry were found (proprietary, open source, emerging etc.), among them most widely used in SoC design is AMBA from ARM which has a de-facto status in majority of SoC designs. We described our experiments with LEON3 above; LEON3 uses AMBA 2.0, so it gave some basic idea of start point of working with AMBA. Some basic investigations to connect eFPGA with AMBA were done which will be briefly described in next section.

Silicon properties: While eFPGA is an FPGA from architectural standpoint, there is an enormous challenge regarding eFPGAs in SoCs which further magnifies the well known weak points of FPGAs. On the top of them is the large silicon gap, we addressed this challenge of FPGAs in chapter 2 (section 2.1.3), but it is interesting to note that this issue is further challenging for eFPGAs, particularly form commercial stand point. The eFPGA in addition to providing/satisfying the flexibility demands of SoC have to provide them in an acceptable silicon budget to justify the value proposition for SoC. It is for such reasons the philosophy regarding eFPGAs in this thesis work has remained to be very small, highly customized (this work is just the first step) and domain specific, used only where flexibility is essential. Furthermore we also investigated some beyond classic approaches to further increase the potential of eFPGAs (MRAMs) which we will briefly highlight in section 5.4.

Silicon tools issues: The integration of eFPGA in the target design/SoC should be easy/flexible using the standard silicon tools flow. The soft technology independent nature of eFPGA also helps in easy integration of eFPGA using standard silicon flows along with the other components in the design.

Fabrication and verification issues: FPGAs are a specialized form of hardware in a sense that although being a straightforward highly scalable digital circuit their particular nature of immense routing often lead to relatively higher amount of metal layers (common in high-end Xilinx and Altera devices) with special process/features provided by Fab. In the case of eFPGA this issue gets further crucial as it is an IP not a separate device/die. An unusual or expensive fabrication demand of eFPGA from fabrication point of view

compared to all the other elements (IPs) in the SoC can lead to tough decisions for the SoC vendor. The eFPGA architectures that we explored in this work are compact, light weight and can effectively route in 5 metal layers on a standard 7 metal layer ST process. A test chip on 65nm (see end of chapter) was also made to prove and verify the architecture (not direct contribution of this work) and *is fully functional*, hence making it effective for integration without any barriers.

Verification (beyond scope of this thesis) is a significant challenge for eFPGAs. It is often the hardest part (getting harder at every new node) in chip design, is particularly special for programmable devices like FPGAs and add 1 bar of additional challenge for eFPGAs.

5.3.2 Basic experiments with AMBA

In continuation of the work with LEON3 [S-7], some basic experiments were done to integrate the eFPGA with LEON3 through AMBA (AHB and APB) and create a basic platform to conduct experiments. However no exhaustive work like [S-7] was done and it is among the future perspective of the thesis to create a full infrastructure following the industry standards for IP integration.

Figure 5.9 shows the basics of the work done which emerged using the knowledge of LEON3 experiments and the challenges regarding system integration. A low gate count wrapper of flip-flops around the IOs of eFPGA was created in the form of four banks. The Banks Controller manages the communication of data in and out of eFPGA through the AMBA bus which is controlled by software from LEON3 making it very flexible to conduct experiments. The Banks also provide a buffer for silicon tools by isolating the eFPGA from the timing paths from rest of the SoC, facilitating the timing closure by silicon tools. Hence this basic infrastructure provided an easy integration; good software control over eFPGA for making experiments (helpful in verification also).

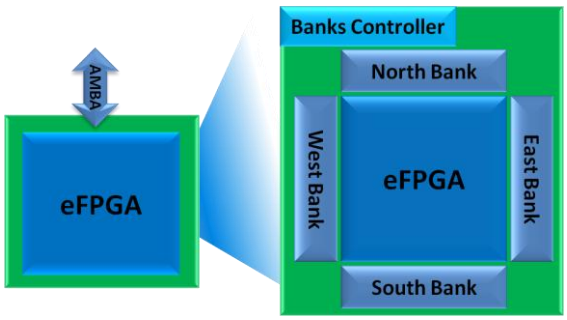


Fig. 5.9: Basic experiments for eFPGA communicating with AMBA (AHB and APB)

5.3.3 Processor inside FPGA vs FPGA inside Processor perspectives

At this point one very important factor "*Processor inside FPGA vs FPGA inside Processor*" can also be briefly analyzed in the light of survey research, experience from experiments conducted; to visualize potentials of eFPGAs in the scenario. Historically one has largely failed compared to the other although theoretically being quite same. When we try to put an FPGA inside a processor (functional unit) some crucial problems arise which have often been overlooked or ignored in many previous works. Despite it is the best and most optimal way to get the reconfigurable accelerator, *it suffers at least three major flaws/cons* which makes it un-acceptable especially for industry in most cases [S-3].

- It changes the processor integrity both hardware and software (processor vendors including end customers do not like that)
- Easy said than done, it is complex to find an effective way of connecting it to the processor data path and estimating the best size and granularity of the reconfigurable portion

- In principle it makes the processor guy the solution provider, who in addition to compete and innovate in two different domains (due to that) has also to adapt his processor for every change needed due to reconfigurable portion

The end result in most cases is chaos, it is such and several other reasons that all leading processor companies and SoC vendors usually do not like to hear or create such solutions. One must not confuse with the custom processors like Tensilica, ARC etc. (successful in industry) in the current discussion scenario; they are custom to a target, not reconfigurable data path to adapt to a target.

Going the other way around makes the story quite different and is the reason why processors inside FPGA devices (soft/hard) have enjoyed huge success and are now going to next level (software centric design with hard processors, as discussed in chapter 2). It makes things lot easier and sound both technically and commercially in comparison to the other way around approach. The processor IP provider fully concentrates on his product differentiation and does not have to make a modified IP for every end customer and due to that all end customers also take the benefit of software dominance of that processor. The FPGA companies based on their target market create products which combine the benefits of software (standard processor) and their reconfigurable hardware. The complex functions can be moved to the reconfigurable part and act like an IP to the processor; furthermore ESL is emerging for helping auto transform the software to hardware conversion for software developers.

The concept extends for SoC providers also; they can take leverage of same concept to differentiate their products by using in house or 3rd party eFPGAs, a die from FPGA vendors (SIP) and connect it as a peripheral with standard interfaces. All players fully focus on differentiation of their IP/Product/SoC. Figure 5.10 illustrates the three scenarios (trilogy of Reconfigurable Accelerators).

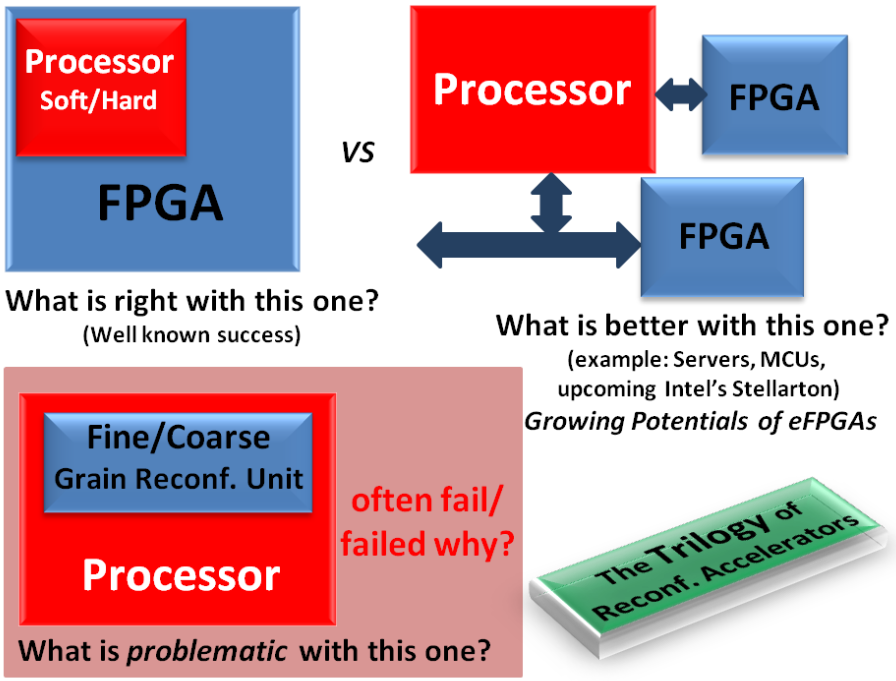


Fig. 5.10: The trilogy of Reconfigurable Accelerators (problem of reconfigurable functional unit)

5.4 MRAM based eFPGAs case study

This section briefly outlines perspectives of MRAMs for eFPGAs. We observed in chapter 2 that MRAMs are among prominent beyond classic emerging elements which in addition to their candidacy of universal memories have some distinguishing properties that can be exploited particularly for FPGAs. Section 5.4.1 describes basic fundamentals of MRAMs, about their working principle, the hybrid CMOS hardware, some basic properties of MRAMs compared to SRAM and FLASH are described, and different types of MRAMs and their properties are discussed. Section 5.4.2 explains the perspectives of MRAMs for eFPGAs in the light of survey research and conducted experiments of this thesis work. Section 5.4.3 highlights the taped out test chips to validate the concepts.

This section is mostly based on the joint research work presented in [S-4][S-5], the MRAM detailed layout works, simulations and test chips are not contribution of this thesis work, this work using the survey knowledge of industry, FPGAs and conducted experiments helped to provide eFPGA architecture, tools and perspectives for MRAM based eFPGAs.

5.4.1 MRAMs fundamentals

This section provides the basics of MRAMs based hybrid CMOS circuit for configuration cell. The basic properties of MRAMs compared to FLASH and SRAM are described and comparison of different types of MRAMs is presented. The deeper details are beyond the scope of discussions, only the fundamentals are discussed for providing the general overview and ease of discussions for perspectives.

1- Working principle and Hybrid CMOS hardware

In the MRAM technology, storage information is not controlled by electrical charges (as in EEPROM or FLASH) but by a change of resistance of a magnetic nanostructure (the magnetic tunnel junction (MTJ)). A MTJ consists of two ferromagnetic layers separated by a thin tunnel oxide layer, one of the ferromagnetic layer's magnetization is pinned (fixed), while the magnetic orientation of the second layer (free layer, soft layer or storage layer) is free. The resistance of the MTJ is either low (Rp) or high (Rap) depending on the magnetization orientation of the free layer relative to the pinned reference layer, parallel or anti-parallel. The resistance variation is characterized by the TMR (Tunneling Magneto-Resistance) ratio that is defined by the TMR = (Rap–Rp)/Rp, currently the TMR is about 150% [Crocus, 5.16].

Thermally Assisted Switching (TAS) approach combines a local heating of the junction (allowing discriminating the junction that has to be written) and a single low amplitude magnetic field. This writing method also requires several steps that are depicted in figure 5.11. When the junction is heated above the blocking temperature (~150°C) by a current (Iheat: ~270µA) flowing through the junction, the magnetization of the ferromagnetic layer is freed and can be reversed under the application of a single low amplitude magnetic field. The magnetic field is maintained after the heating current is released so that the junction cools down under magnetic field to ensure that the magnetization stored in the free layer is pinned correctly. We can see the write-line on which a write current pulse is applied. The magnitude of this current must be high enough (Iwrite: ~8mA) such as the generated magnetic field is larger than the coercive magnetic field. For the heat, switch and field cool steps the programming timing could be realized in less than 35ns [Crocus, 5.16] (for brief comparison SRAM : < 10ns, FLASH : > 150µs).

For the design of hybrid circuits CMOS/Magnetic, it will be necessary to transform the magnetic information into electrical information, to achieve that a hybrid circuit designed is shown in figure 5.12 [Guillemenet08, 4.58]. The structure of the TAS-MRAM cell (figure 5.12) consists of cross-coupled inverters (MN1 & MP1, MN2 & MP2), two MTJs for a non-volatile storage with complementary values to unbalance the latch during a read step. The figure also illustrate that (will be further explained later) MRAMs are stacked above the CMOS in higher metal layers with no physical link with silicon substrate of CMOS. The writing line is implemented in a "U" shape such as allowing writing the data and its

complement in the two MTJs of each MRAM cell by providing opposite magnetic fields H1 and H2. Two transistors MP3 and MP4 are driven on their gates by a signal "*sense*", which act as "isolation" transistors (to preserve the data stored is the latch during a write step) and selection transistors MN3 and MN4 are used to enable the heat operation. This structure operates during the read mode (writing CMOS latch) as follows: When the signal "*sense*" is at low logic level, transistors MP3 and MP4 are switched on allowing to pass a read current across MTJs until the latch. The intensity of these current are both different (Imin and Imax) because these MTJs are in opposite state (Rmax and Rmin). In consequence, a differential potential is generated at the boundary of the latch (Vmin and Vmax) unbalancing it. At this moment, the latch is in a meta-stable state. Then, when the signal "*sense*" goes from the low to the high logic level, transistors MP3 and MP4 are switched off, depriving the contribution of energy in the latch to maintain it in the meta-stable state. The structure is going to fall over thus automatically towards the closest stable state (in this case *Vout* ="1"). The total read cycle is about 1ns [Guillemenet08, 4.58].

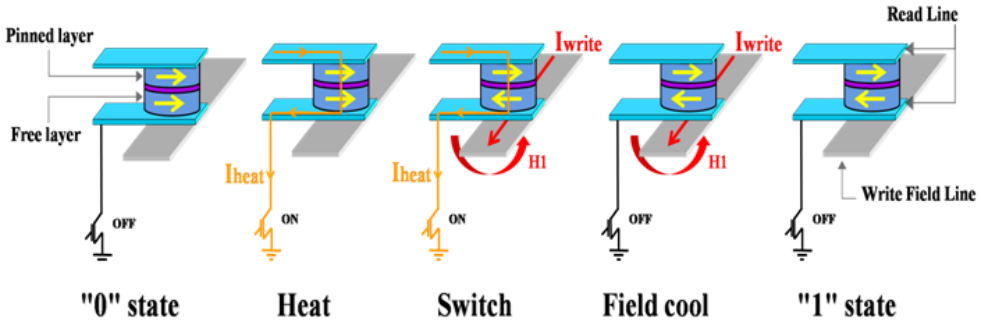


Fig. 5.11: Step sequences of TAS write operation in MTJ

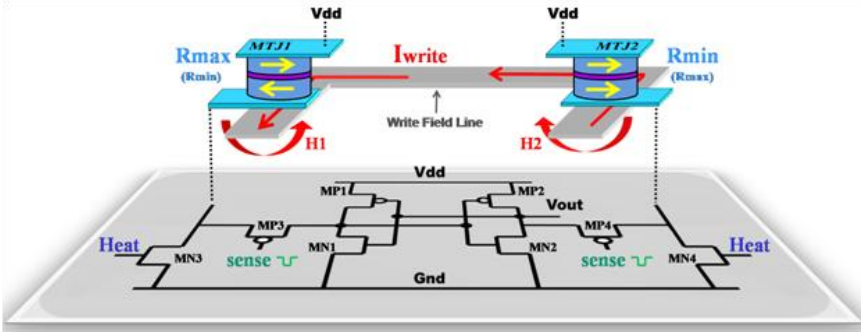


Fig. 5.12: MRAM transistor level Hybrid CMOS circuit

	MRAM	SRAM	FLASH
Read Speed	FAST	FASTEST	MEDIUM
Write Speed	FAST	FASTEST	LOW
Cell Area	LARGE	MEDIUM	LOW
Scalability	GOOD	GOOD	LIMITED
Endurence	Medium	GOOD	LIMITED
Non-Volatility	YES	NO	YES
Low Voltage	YES	YES	NO
Rentension (years)	10	0	10
# of additional mask	4	0	12
Radiation hardness	++	--	--

MRAM Types	FIMS	TAS	STT
Read time(ns)	< 30	< 30	< 30
Writing current (mA)	~20	~8 Heat current: ~0.34 (120nm)	~0.2
# additional masks	5	5	5
Endurance (cycle)	10^{12}	10^{6}	10^{12}
Retention (year)	10	10	10

Fig. 5.13: Basic Properties and types of MRAMs

2- Types and Properties

As stated before MRAMs are among the candidates of universal memories [5.20][5.21], figure 5.13 provides an overview of MRAMs compared to mainstream memories (DRAM is not discussed as we are focused on memories in use in FPGAs configuration). The table illustrates how MRAM compares in

different aspects to SRAM and FLASH. The table on the right hand side explains the properties of different types of MRAMs [Guillementet10, 4.60]. The major difference between different types is the writing current. The **FIMS** (Field-Induced Magnetic Switching) is among the first one proposed. It has a high write current which penalizes the silicon area as large transistors and metal wires are needed to support that high current. The **TAS** (Thermally Assisted Switching) that was used in our works improves this issue. Finally the ***STT*** (Spin Torque Transfer) is under major focus of industry at present which further improves the write current issue significantly. As STT is same like TAS from fabrication stand point all our work is upgradable to STT, leveraging its superior benefits. Next section discusses the perspective of MRAMs for eFPGAs.

5.4.2 Perspectives for eFPGAs

This section provides the perspectives of MRAMs for eFPGAs. We have classified them into five categories to highlight the distinct benefits which MRAMs bring compared to classical solutions. They are thematically illustrated in figure 5.14 concept diagram in form of five major distinctions (5ive Stars of MRAMs) which are discussed below.

1- Re-Programmable Non-Volatility

Having a non-volatile solution for FPGAs is highly attractive and often desired. The major motivations for that are well known. It provides a single chip solution which is instant-on (eliminates need of external non-volatile memory). For eFPGAs these advantages become further prominent. The instant-on feature is very attractive from power point of view; eFPGA can be shut down (fully/partly) when not used, to save power which is a crucial challenge at present. In addition to having non-volatility it is also desired that the non-volatility is multiple time programmable (MTP) compared to one time programmable (OTP) only. At present FLASH is the leader in this area, but as we discussed in chapter 2 [S-5] there are some downsides with FLASH where MRAMs provide superior benefits (illustrated in figure 5.13 also). We will discuss it further in the fabrication section below shortly.

2- Radiation Hardness

As the storage of information in MRAMs is magnetic compared to electric, it provides enhanced radiation hardness ability to MRAMs based solution compared to classical ones. Radiation Hardness (beyond scope of this thesis work) is an important requirement in critical applications (military, aerospace etc.). MRAMs based eFPGAs in addition to being non-volatile also achieve superior characteristics for radiation immunity. [Cargnini&Guillemenet10, 4.61] describes how MRAMs can help in this regard compared to conventional SRAM based solutions.

3- Shadowed Dynamic Reconfiguration

Dynamic Reconfiguration is an interesting aspect of FPGAs which helps to further exploit the programmability benefits of FPGAs for several dimensions for instance increased area and power efficiency by reusing the full or portion of the device. We discussed in section 5.1 in general discussions about the status and growing potentials of dynamic reconfiguration. For eFPGAs dynamic reconfiguration can be further interesting compared to device FPGAs. For instance, as an eFPGA is small IP in system it can be frequently/based on needs reconfigured to perform different tasks (MRAM are re-programmable and much faster to write compared to FLASH). Also as eFPGA is a customizable IP, architecture with more dynamic reconfiguration friendly configuration system can also be designed based on target needs. Furthermore MRAMs based hybrid CMOS configuration provides a unique opportunity for dynamic reconfiguration i.e. Shadowed Dynamic Reconfiguration (on the fly). Figure 5.15 illustrates the concept; we explained the basic operation of hybrid CMOS cell above with which the figure can easily be described. As the central latch (cross coupled inverters) is isolated from the MRAM portions by the transistors controlled by “sense signal” we can write new value to the MRAMs while the device is running (latch is providing the configuration to device), this is shown in figure in three steps/phases. We can see

that in phase-b we upgrade the value of latch from phase-a MRAM values and in phase-c we write new values to MRAM without disturbing the value of the latch which can be switched to new value any time like was done in phase-b. This shows how MRAMs can provide a remarkable benefit for dynamic reconfiguration by virtually providing an inherent dual-context solution with same base hardware. Similar characteristics with conventional techniques will require additional transistors.

4- Multi-Context (with low silicon overhead)

Multi-Context in some aspects is an enhanced form of Dynamic Reconfiguration. The concept is very impressive as it utilizes the profound benefit of the programmable logic capability of FPGAs. By having more than one layer of configuration allows rapid switching to a new functionality (ultra high speed dynamic reconfiguration). However achieving such features come at a heavy silicon price and make the value proposition of multi-context weaker compared to tradeoff for most cases, at present no commercial FPGA support multi-context (we discussed Tabula solution in chapter 2 but that is relatively a different style of architecture). We described above the unique benefits of MRAMs for dynamic reconfiguration, they also provide superior features in regards of multi-context compared to classical solutions. Figure 5.16 highlights the scenario. It is intuitive to understand the multi-context hybrid architecture principle and working from the discussions above. We can see that compared to classical solution of double the configuration plus multiplexer overhead for the case of MRAMs we only need to add a pair of MTJs to add a new context which is clearly more silicon efficient. Furthermore from our discussions of Shadowed Dynamic Reconfiguration above it must be noticed that each context inherently has a virtual dual-context. Adding physically more contexts on one hand will enhance direct instant switching among different configurations, and on the other hand also provides potentials of clever exploitation of shadowed dynamic reconfiguration among multiple contexts to further enhance the capabilities of eFPGA for instant switching to different configurations (particularly interesting for a co-processing scenario shown in figure 5.14). The table on bottom compares the transistor count for MRAM based vs conventional solution. We can see that how rapidly the transistors increase with increasing context for conventional solution compared to MRAMs solution.

Con of MRAM: The table also depicts that a single MRAM configuration cell is more expensive than a classical 6T SRAM, this issue is significant from device FPGA stand point if a target demand is conventional with no multi-context and dynamic reconfiguration (often the case in reality). Also from power stand point if the nature of target application domain does not allow exploitation of instant-on/shutdown potential it is trivial to observe that MRAM based cell will have more static power compared to SRAM (more transistors leak). This provides research motivation for finding more compact memory cell architectures (are already underway by research community).

5- Integration and Fabrication ease with conventional CMOS

All the benefits and potentials that we have been discussed above can easily be nullified if the MRAMs cannot/too hard to integrate and fabricate with the rest of the design. For eFPGAs this issue is further significant as complex demands of just one IP can make the SoC level decisions and silicon budgets complex and can force a revisit to the value proposition of that IP in global scenario. In such regards also, MRAMs have promising advantages. Figure 5.17 illustrates the integration and fabrication ease benefits of MRAMs. The principles are explained based on our work done on MRAM based FPGAs between Menta and Lirmm (University of Montpellier) however it explains the fundamental benefits of MRAMs in practical scenario of industry. The figure on the left hand side shows how the configuration cells of hybrid CMOS are integrated with the architecture of eFPGA (in future they can be directly available from Fabs). We can see that all the flows are based on standard hardware design flows and using the classical well known state of art silicon tools.

The figure on the right hand side shows the fabrication flow where several interesting observations can be made. The first and most important point is that the MRAMs are integrated (stacked) on the top of

conventional CMOS process. This is a significant benefit from both technological and commercial stand point. We briefly covered this issue already in chapter 2 discussions [S-5]. Having the MRAMs on the top allows using conventional CMOS process, taking leverage of the latest node (difficult with FLASH [S-5]) where MRAM can still be of some different node (already in our experiments we used 130nm CMOS and 120nm MRAM, in principle other way around is also possible). Furthermore MRAMs require fewer mask levels compared to FLASH (figure 5.13), giving further benefits from cost point of view.

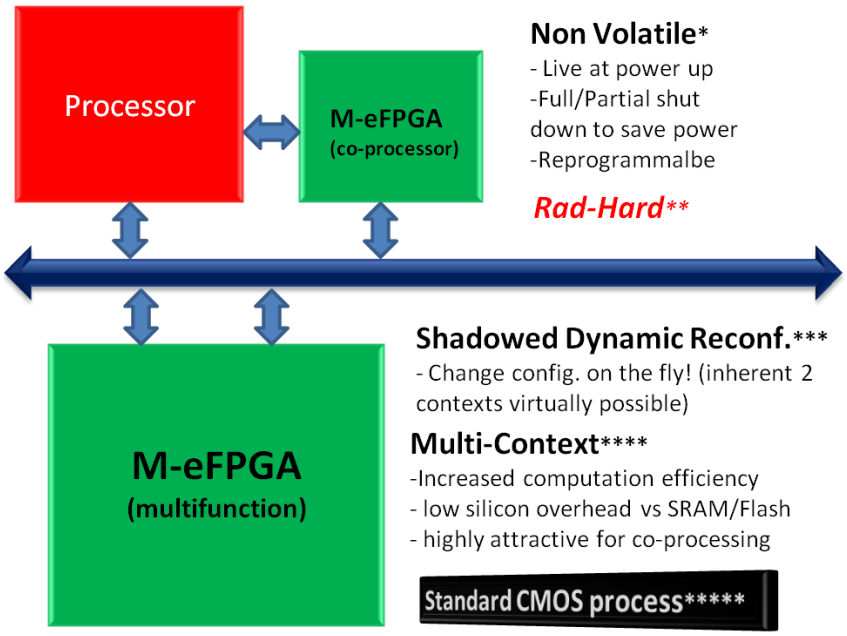


Fig. 5.14: Perspectives of MRAMs for eFPGAs (the 5ive stars)

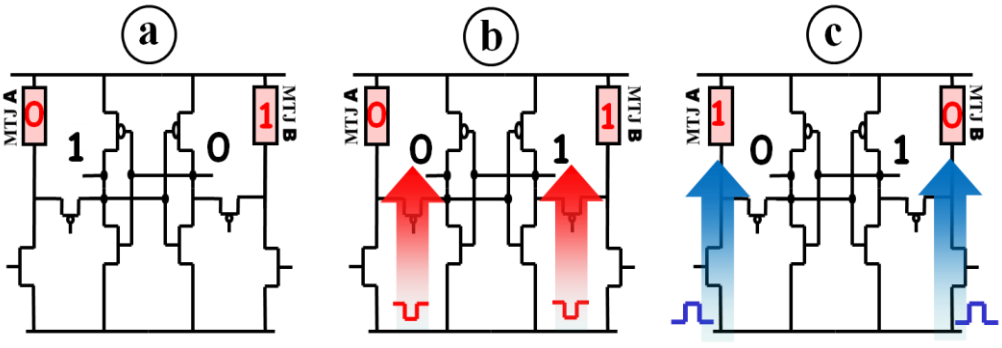
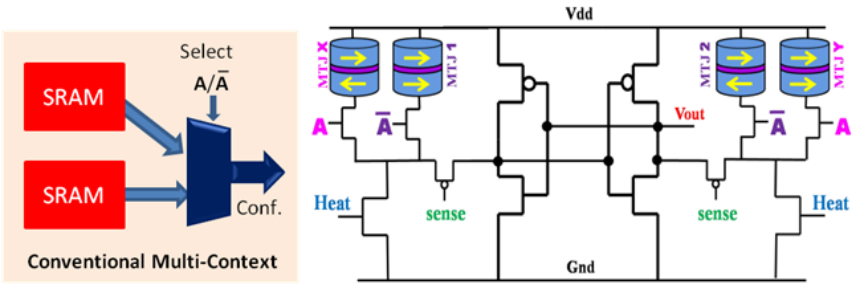


Fig. 5.15: MRAM based Shadowed Dynamic Reconfiguration



No. Context	1	2	4	8	16
Transistors: SRAM+MUX	6	16	34	68	134
Transistors: MRAM	10	12	16	24	40

MRAM based Multi-Context (low silicon overhead)

Fig. 5.16: MRAM based Multi-Context configuration cell vs classical Multi-Context cell

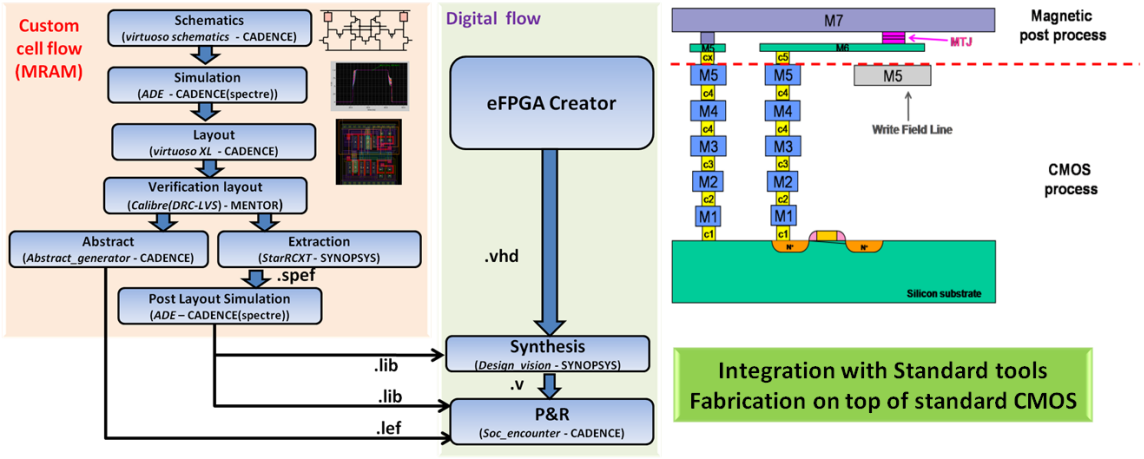


Fig. 5.17: MRAM integration and Fabrication with standard tools and standard CMOS

Test Chips (MRAMs, Latch/SRAM based configuration)

This section provides brief details of the test chips made in close collaboration with this thesis work which utilized the eFPGA architecture (fully/partially modified to meet test chip issues) that was explored in this thesis (chapter 4.1).

MRAM based configuration Test Chips (ST130nm)

MRAMs still have a way to go before they become main stream like FLASH/DRAM but the interest and roadmaps of industry is positive enough to claim that they are becoming a reality now and are not just a research project anymore [S-5][5.17][5.18]. The test chips taped out by LIRMM and Menta in ANR (French National Research Agency) project [EEtimes09, 4.78][EEtimes10, 4.79][S-4] are a living proof of that. To validate the research concepts and potentials of MRAMs discussed above some test chips are taped out on 130nm (130nm ST CMOS + 120nm Crocus MRAM). This further distinguishes the years of research of LIRMM on MRAMs for FPGAs to be among world's first known MRAM based FPGA test chips tape outs. The test chips are not direct contribution of this work and deeper details are beyond the scope of discussions. The fundamentals are provided here for the completion of discussion for advocating the above mentioned perspectives which will become easy to exploit and manufacture as MRAMs continue to get evolved and matured.

Figures 5.18 and 5.19 show two taped out test chips for validating MRAM based FPGAs on 130nm process [S-4] (130nm ST CMOS, 120nm Crocus MRAM, MRAM post processing by CEA-LETI [5.19]).

The first one is a small 2x2 architecture with 4 LUT-4 connected with a routing architecture. The second one is a large FPGA with 1444 LUTs (LUT-4, Cluster-4, Channel-16) on which practical applications can be mapped for experimentation. The test chips are under experimentation for functionality.

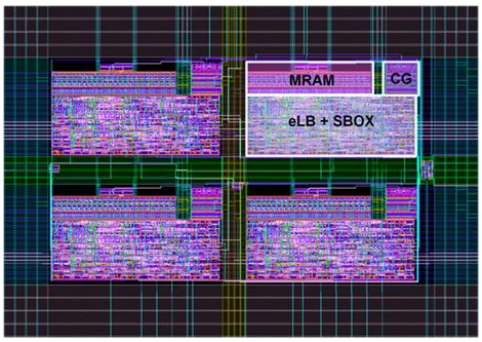
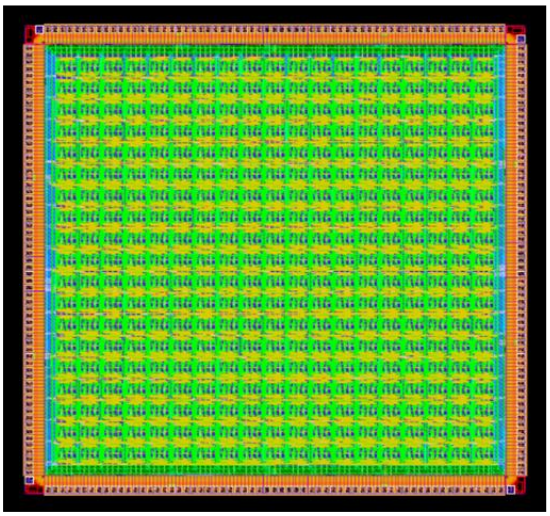


Fig. 5.18: MRAM 130nm small 4 LUT-4 Test Chip (ST CMOS + Crocus MRAM)



# LUT-4	1444
# Tiles	361 (19x19 array)
# Sequential elements for the user logic	1444
# of MTJs	187 720
# of Transistors	8 534 000
Silicon area (130nm)	20,3 mm2, (Core: 15.64mm2)
MRAM Reconfiguration Tile Energy	9,01 nJ
MRAM Restoration Tile Energy	25,46 pJ
Clock Frequency	100 Mhz
Full configuration time	72µs + 93860 clock cycles
Tile reconfiguration	200ns + 260 clock cycles
Input/Output	76 In/ 76 Out

Fig. 5.19: MRAM 130nm 1444 LUT-4 FPGA Test Chip (ST CMOS + Crocus MRAM)

Latch/SRAM based configuration Test Chip (ST65nm)

Figure 5.20 shows a 1mm^2 (core= 430 um^2) test chip on 65nm LPSVT process. The chip has 16 tiles (LUT6, Cluster-4, Channel-16), 12 of them use Latch-based configuration (pure soft) and 4 use SRAM-based configuration (partially custom). There are 32 pads. The chip has been verified for functionality and deeper experimentation is under way.

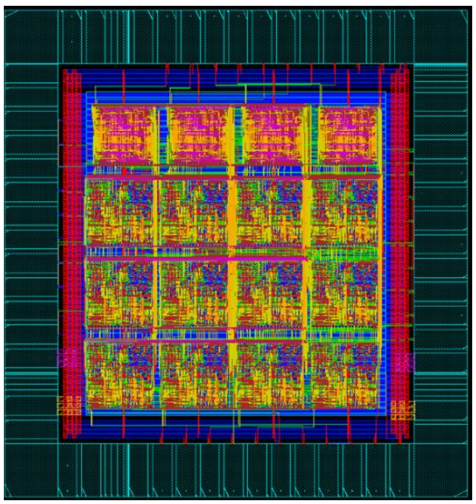


Fig. 5.20: ST65nm eFPGA Test Chip of Menta (64 LUT-6)

5.5 Summary

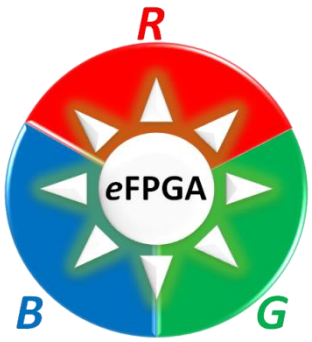
The chapter presented the contribution G (chapter 1) of this thesis work to explore the challenges and potentials related to systems perspectives for eFPGAs. Section 5.1 provided a general overview of the potentials of eFPGAs for SoCs in the light of changing dimensions in industry. It revisited the potentials of reconfigurable accelerators and highlighted the prospects of eFPGAs in the spectrum with their leverage of standards for programming, IP eco-system, benefits of ESL exploitation, flexible hardware. Section 5.2 addressed in detail reconfigurable acceleration experiments with LEON3+eFPGA with focusing on programming complexity by exploring ESL use and detailed silicon tradeoff analysis to visualize a more realistic picture for the challenges and potentials for eFPGAs in this regard. Section 5.3 discussed the challenges regarding system integration for eFPGAs in the light of survey knowledge and conducted experiments. It also briefly addressed the "Processor inside FPGA vs FPGA inside Processor" perspectives and tried to highlight the possible reasons (that have often being ignored by researchers) for high failure of reconfigurable data path like solutions in industry based on survey knowledge of this thesis work. Conventions are not enough to move ahead, section 5.4 highlighted the perspective of beyond classics MRAMs for eFPGAs which can significantly help augmenting the potentials of eFPGAs in future SoCs. An overview of test chips was also given which makes the explored architecture of this thesis work silicon proven for both conventional and beyond classic technologies.

The key points observed/discussed can be summarized as follows.

- eFPGAs can provide promising potentials for product differentiation, time to market and multitude of other benefits for SoCs
- The soft, technology independent eFPGAs provide instant portability to any process, removing the fab/node dependent solution barrier. If ROI justifies a hard solution can also be made
- The reconfigurable acceleration can help in growing challenges of industry regarding performance and power walls. A revisit to the potentials of Reconfigurable Computing is needed to leverage its benefits and find solution for the past mistakes in standard industrial way
- The rise of ESL and FPGA-like hardware of eFPGAs can help bringing FPGAs benefits to SoCs
- eFPGAs can provide attractive solution as a reconfigurable accelerator, experimental investigations were conducted with a standard processor (LEON3) with full silicon tradeoff visualizations
- ESL has a promising future for SoCs, FPGAs and eFPGAs
- eFPGAs must follow an industry standard (like AMBA etc.) to be integrated with systems, they must have attractive silicon properties (area, power, speed) for a value proposition for SoC. They must be easy to integrate using standard tool flows and should be easy to fabricate (no special requirements)
- The beyond classic MRAMs can provide salient benefits like non-volatility, radiation hardness, shadowed dynamic reconfiguration, multi-context and ease of fabrication and integration etc. Such features can help highly augmenting the potentials and value proposition of eFPGAs for SoCs

The next chapter concludes this thesis work and provides future outlook.

Chapter 6: Conclusions & Future Lines of Research



This chapter concludes the thesis work, explains the summary of contributions, knowledge gained and provides the possible future lines of research.

6.1 Conclusions

This thesis extensively revolved around embedded FPGAs (eFPGAs). It tried to investigate industry for its changing trends, crucial challenges it is facing at present where programmable devices have strong potentials and how/where/if eFPGAs fit in that spectrum. It conducted a survey of programmable technologies for their potentials and status in industry and tried to explore why FPGAs dominate/historically dominated in that area compared to other related/competitive approaches. In-depth study of FPGAs was made to understand their research challenges for creating FPGA-like embedded FPGAs. In the light of survey knowledge, the concept and motivation of soft eFPGAs was presented with standard RTL programming flow and rich tools infrastructure to perform architectural exploration of eFPGAs. Detailed architectural explorations were performed to create customized silicon efficient eFPGAs (area, power, performance). Since the scenario of eFPGA is incomplete without systems; efforts were done to investigate challenges, perspectives of system integration for eFPGAs, practical experiments in the direction of reconfigurable acceleration with standard processors were conducted. To further enhance the potentials of eFPGAs for systems, investigations in beyond classical technologies and methods were also explored and an overview of perspectives of emerging MRAM memories in this regard was presented.

Contribution Axes: The thesis contributions were classified in three major dimensions/axes in the introduction (chapter 1) namely; R (Industrial Survey & Analysis), B (eFPGA Architectural Explorations) and G (eFPGA in Systems), the entire thesis was tightly coupled to them in the form of chapter organizations and discussions. The joint knowledge of these three axes was addressed as an auxiliary fourth axis in the form of tools suite (eFPGA Creator) which facilitates creation of efficient eFPGAs. The following sections describe brief overview of the contributions done and gained knowledge which helps to define future outlines.

6.1.1 Summary of Contributions

The summary of contributions of the thesis work is briefly described below in form of chapters.

Chapter 1 provided the global overview and challenges of industry/academia at present, presented motivations, objectives and key contributions of thesis, and graphically bounded them to key dimensions facilitating flow of thesis discussions.

Chapter 2 investigated in detail the fundamentals of FPGAs, eFPGAs and a general survey particularly focused on programmable technologies. It covered in-depth analysis of FPGA research challenges and advances in state of art and academic research. It investigated previous efforts in industry and academics for eFPGAs or similar directions. An extensive global survey of semiconductor industry was presented (perhaps first one of its kind), analyzing different markets and their status. It specifically focused on programmable technologies and provided a qualitative and quantitative analysis of different types of them with their market position and potentials. It discussed in detail the potentials and strengths of FPGAs and versus FPGAs technologies to discover reasons of FPGAs dominance. The chapter provided detailed motivations and guidelines for the work done and presented in the thesis.

Chapter 3 provided an overview of the architecture fundamentals of eFPGA explaining the motivations behind them, particularly in soft eFPGA scenario. It briefly outlined the CAD tools (eFPGA Programmer), which provides the standard RTL programming flow. It discussed in detail the philosophy of eFPGA creation and exploration tools suite infrastructure (eFPGA Creator) and explained the efforts done to create these tools which provide a user-friendly infrastructure for architectural explorations on real process nodes.

Chapter 4 extensively addressed the eFPGA architectural explorations which are the biggest contribution of this thesis work in terms of time spent on experiments, gathering knowledge and infrastructure to perform those experiments. It presented the exploration in two levels. First level presented in depth general explorations for nearly all fundamentals (LUT size, cluster size, channels, routing architecture challenges and also growing importance of power etc.). Second level conducted deeper levels of customizations for further enhancing the silicon efficiency (area, power, speed) of eFPGAs while maintaining good routing efficiency. It investigated how the routing efficiency of eFPGA can be enhanced by connecting the logic block in diverse ways to less flexible silicon efficient routing architecture and found promising directions of new kinds of architectures (with eliminated conventional connection block and local interconnect) which compare/rival past published (VPR/VPR-like) research attempts. It also briefly provided some near term directions of enhancement of achieved results and observations to wider design space exploration.

Chapter 5 addressed eFPGAs in systems potentials, challenges and perspectives. It provided a general overview of the challenges of industry where eFPGAs can play interesting part; a particular emphasis was put on reconfigurable acceleration in the light of benefits that eFPGAs can bring with their specific properties (standard programming, high flexibility, IP-leverage etc.) reviving motivations of several similar past efforts that unfortunately failed. A practical exploration of reconfigurable acceleration scenario with standard processors was investigated and presented with focus on programming complexity and silicon tradeoffs visualizations. The challenges regarding eFPGAs for real-life system integration were highlighted. View point for possible reasons of several failures of reconfigurable data path like past efforts was presented in industrial scenario. Conventions are not enough to move ahead, perspectives for beyond classics potentials of MRAMs for eFPGA was presented. Brief overview of test chips taped out based on eFPGA architecture of this work was provided which makes this work silicon proven on conventional (65nm CMOS) and beyond classics (130nm CMOS+MRAM).

References section was created as an interactive stand alone chapter with categorized bibliography.

6.1.2 Knowledge Gained

Some key aspects of the general knowledge gained from the contributions are briefly described below that were discussed in detail in the thesis report and corresponding contributions [S].

***Semiconductor Industry*:** Extensive study of semiconductor industry was done including beyond technology commercial aspects of industry (beyond scope of thesis discussions). A detailed survey focused on programmable technologies was conducted to explore potentials and status of different technologies including investigations to understand reasons of failures of several past efforts. The potentials and challenges for eFPGAs in changing trends in industry were investigated.

***In-depth knowledge of FPGAs*:** Since the research conducted in this thesis work focused on FPGA-like eFPGAs, an in-depth study of FPGAs was done. The thesis investigated scientific and commercial literature of the state of art, studied patents to observe real-life challenges and how state of art deal with it, performed detailed analysis of academic research. Extensive practical experimentations were done to investigate eFPGA (in principle FPGA) architecture.

***Importance of Standards*:** Standards play a vital role in industry. From commercial aspect can even monopolize some technologies/companies. Setting aside the commercial aspect, the complexities of chip

design and related surrounding eco-system has gone so high that it is often very hard for non standard technologies/innovations to succeed finding a niche in industry despite they are offering some highly innovative solutions to key challenges. This brings the high challenge to scientists both in industry and academics to clearly understand the challenges and devise effective roadmaps that pave the way for new innovations (protection of innovation from dying) that have strong potentials to get standardized in a longer term or provide strong competition to existing solutions which bring industry to inflexion point of tough and revolutionary decisions. As an example there is a lot to learn for researchers from the FPGA vs Coarse Grain story, and at present slow and steady move of industry to multicore in different dimensions.

***Challenges of Power Consumption*:** Since Moore's law crossed 90nm node, power consumption has become a paramount issue and challenge throughout the industry and has shaken almost four decades of science and business legacy. Power consumption has become a particular research and commercial challenge for FPGAs also, against competitive technologies.

***Challenges for eFPGAs*:** The eFPGAs have natural undeniable potentials and several attempts have been made both from industry and academics in past (chapter 2), however it is well known that hardly it ever succeeded in industry. Considering eFPGAs technical challenges, following key things are worth noticing which were discussed and some were addressed in the thesis.

- **Tools:** Play a vital role and usually had been main reason of failures of many. They should be high quality and must provide some standard flow (RTL or ANSI C/C++ etc.). RTL is more interesting for FPGA-like eFPGAs as it provides dual exploitation benefits (ESL) and IP eco-system leverage
- **Silicon Gap:** FPGAs are well known for their large silicon gap with ASICs, this issue gets further prominent for eFPGAs. The eFPGA must have attractive logic density in acceptable power budgets and desired performance requirements of target ASIC/ASSP/SoC to be an interesting value proposition
- **Hard vs Soft:** Is related with the silicon gap issue. FPGAs (eFPGAs) have a specialized class of hardware which can be better implemented in full custom (like commercial FPGAs) to have as silicon efficient solution as possible. However the target scenario of FPGAs and eFPGAs is (can be) very different. FPGAs are devices that are mass produced for a specific fab node. eFPGAs process node is determined by target device. In deep submicron nodes creating full custom designs have become extremely challenging and expensive even for a single node. Covering a broad range of fabs/nodes is virtually impossible (ROI issues). Soft technology independent customized eFPGAs of small sizes can bridge this fab/node dependence gap in different scenarios. *Something is better than nothing*!
- **Integration:** An effective and standard integration both in terms of silicon tools and industry standard IP interfacing (AMBA, NoC etc.) with the SoC is crucial
- **Customization:** A rich infrastructure which facilitates creation of target dependent customized efficient eFPGAs for SoCs with good value proposition for all above mentioned challenges

***Potentials of Beyond Classics*:** There is a limit of conventional research of FPGA architecture to address the challenges of FPGAs which has nearly two decades of history of innovations. There are some emerging technologies (MRAMs, MEMS, 3D-stacking etc.) and newer communication techniques (packet switching etc.) that have been neglected or partially explored by researchers that can provide some good potentials for FPGAs and are now gaining attraction among researchers in both academics and industry.

***Industry is heading for platform collision*:** With the pressure generated by power consumption and ever increasing design complexity and costs of newer process nodes, several technologies which were relatively quite different or nonexistent in the past are heading for similar markets that will/may lead to collision in future [S-3] (chapter 2). eFPGAs will/can play an important technical and commercial role in the colliding future (of Heterogeneous MPSoCs, FPGAs, MPPAs etc.).

***eFPGAs Research Outcomes*:** The research conducted in this thesis explored architectures, potentials and challenges for soft eFPGAs with RTL programming flow. It was found that eFPGAs of good silicon

properties were achieved with architectural exploration. Since all investigations were done on real silicon process (65nm for most part) the results presented show realistic properties that can be exploited in SoCs, some basic analysis for eFPGA in SoCs were also conducted to illustrate the potentials. It was also observed that the concept of pure soft eFPGAs helped to exploit proven standard cell libraries and state of art silicon tools to implement the architecture on target nodes with silicon properties (logic densities up to 400+ LUT4/mm^2) that compare/rival basic automatic layout generation published works in academics. However there is still a long road to go to have competitive industrial level properties (the actual beyond thesis target of eFPGA of this thesis work). The next section provides future outlines of research.

6.2 Future Directions

The thesis work explored and contributed in several directions to investigate the potentials and challenges regarding eFPGAs. In general terms since eFPGA is FPGA-like, nearly all challenges of FPGAs apply to it. The state and advancements of state of art FPGAs are well known, so in an industrial scenario can already provide the road to future challenges from a company stand point (beyond scope of breadth and complexity of a thesis), since in principle any FPGA vendor can make eFPGA /vice versa.

Based on the knowledge gained and experiments conducted in this thesis, there can be several future outlines. Figure 6.1 shows thematically some interesting future innovation areas which are briefly explained below in the form of a chronological order providing limited sense of priority.

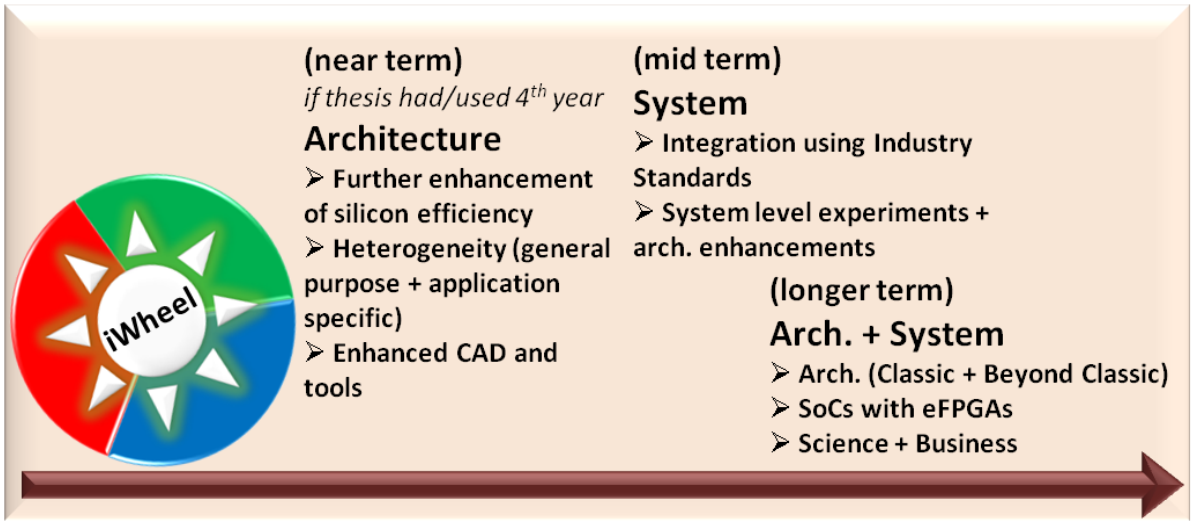


Fig. 6.1: Innovation wheel for future research ideas

Near term:

The most interesting near future extension for this thesis work will be extension of section 4.2.

***Extended Explorations*:** We observed interesting results with significant improvements of silicon efficiency of architectures by SB customizations. Extensive design space exploration in several dimensions will be interesting to explore further possibilities and better architectures. Section 4.2.5 highlighted some exploration directions in this regard.

***Architectural Heterogeneity*:** One of the biggest elements to combat the silicon gap of FPGAs is hard macro blocks. They are highly common in commercial FPGAs, but still are in very early stage in academic research. Hard blocks further help in customizing eFPGA architecture to target needs. It will be interesting to conduct first level experiments for exploring the heterogeneous blocks addressed in section 3.1.2.

***Enhanced CAD and tools*:** Further up-gradation of exploration tools, eFPGA Programmer and eFPGA Creator. An interesting addition to the flow of eFPGA Creator (section 3.3) will be adding an approximate

timing model on some specific node based on the knowledge gained by silicon implementations. This will allow accelerated design space exploration. In current case the silicon implementation phase brings the obvious long delay which is highly penalizing when conducting numerous new experiments. An approximate timing modeling which can help distinguishing and comparing relative gains by architectural customizations will significantly help enhanced design space exploration. Instead of all only set of more interesting architectures will be physically implemented for deeper investigations.

Mid. term:

It will be interesting to investigate enhancement of system integration work and experiments (chapter 5) to standard interfaces (AMBA etc.) to create and explore more real-life scenario. Analyze eFPGA with industrial benchmark applications in system scenario to explore the tradeoffs and find/explore eFPGA architecture enhancements and customization.

Longer term:

In the longer term it will be attractive to investigate more advanced architectural explorations that are competitive to state of the art solutions and challenges of eFPGAs in complex SoCs. Since eFPGA is an IP the investigation of beyond classical methods and approaches will/can be further interesting compared to FPGAs. Chapter 5 presented the perspectives of beyond classic MRAMs for eFPGAs, it will be nice to explore those concepts in detail to find attractive new solutions. **Appendix A3** provides some interesting aspects of using NoCs, MRAMs etc. that will be worth experimenting to explore their potentials for systems (Heterogeneous Reconfigurable MPSoCs) and the fundamental eFPGA/FPGA architecture for routing challenges and heterogeneous hard blocks (IP) integration. The FPGAs are specialized hardware and through full custom design several electrically different implementations of a logical architecture can be made. The thesis work focused on pure soft eFPGAs along with the motivations behind them. However the potentials of custom (hard/semi hard) eFPGAs cannot be ignored. It will be interesting to explore ways to bridge hard and soft eFPGAs. And finally in industrial scenario it requires both technical and commercial contributions for success; it will be interesting to make higher level commercial contributions also for advancement of eFPGAs.

6.3 Concluding Remarks

While the concept of eFPGAs is not new and furthermore does not have an encouraging history, the changing dimensions in industry provide potentials and motivations for revival of eFPGAs, It surely is a missing IP in industry which can play a prominent role in current and especially future SoCs. Efforts must be made to learn from past mistakes and remove the barriers that inhibited eFPGAs success in past. This thesis tried to contribute in this regard in several dimensions. The strong future of eFPGAs in Heterogeneous MPSoCs, Configurable SoCs etc. seems imminent.

Epilogue

The last act of/as CIFRE

PhD in addition to being final (in general!), is a specialized adventure in the university-level education. It is the first time a student actually comes across a complex back-end entity (mostly invisible to outside world) of university, called academia. Which is the foundation/tightly-linked part of research empire and the FAB which tapes-out Doctors and Professors (Cham [5.10] mentioned herein as *one* reference addresses this entity with philosophical depth). Since this thesis work was hybrid (Prologue), it provided an opportunity to independently work/observe in parallel in/with the two distinct worlds [5.11] of "academia" and its complex counterpart "the industry". This perhaps was the hardest thing I had to deal with in my PhD, however in tradeoff, the start-up nature of Menta specifically helped to observe industry with further depth (beyond technology). Having had the power/gift/burden of officially being at the same time "*i*Industry + *i*Academia", helped me to investigate and penetrate these two (part similar yet miles apart) distinct worlds in a way which would have been virtually impossible in any other configuration (pure PhD, pure professional etc.). Tomorrow (logical) I will/have to lose one power (general future/direction of CIFRE PhDs is well known [5.11]), however cherishing memories of the adventures and knowledge gained will surely help/guide in future as they indeed heavily did during the thesis.

As a last act of CIFRE, it brings me to contribute/document some of the top global (unbiased) findings described below in spirit of both academia (we share) and industry (grab differentiation). Those of skill in the areas will instantly recognize that points mentioned are symbolic and can be extended to plurality of points/discussions and real examples (beyond scope/intent of thesis/epilogue). *What I observed/learned/investigated is.*

In one embodiment an entity called Semiconductor Industry:

- We are not primarily doing this for the humanity, it's our business, if we don't do it someone else (competitor) will
- If one can't control/protect one's market share, be visionary enough to foresee future before enemy (needs can/may be created); he will/may be doomed, no matter how big/dominant one is (tomorrow will/may not take care of itself)
- There are usually two ways to do business: 1- Have something that no one has, use/create dominance of/through that; 2- Find weakness of the enemy, use that as strength and crush him with it
- Science alone can/does not run/drive the industry, it takes TWO (geeks + businessmen) to do wonders
- We protect knowledge (patent), exploit/help academia, steal ideas, buy ideas, can/damage science for business, even do miracles of science, are often at high risks (market/economy). All is fair in love & war, damn hard we try playing that!

In another embodiment a corresponding (not entire*) entity called Academia:

- We love: to share, cite and be cited (the beauty & burden of selfless academia)
- There are two distinct kinds of professors in the world: those who contribute science and those who contribute themselves in the name of science. In real-life many do/must/have to use a mixture of it for obvious reasons
- The finest minds of academia (professors) are/made so diversely overloaded that often they don't/can't find fair enough time for what they are/wished/supposed to be overloaded for (homogeneous academia vs heterogeneous industry)
- IEEE/ACM paid downloads business model is/has gone outdated in/for internet era: Google business model supremacy
- Only ***P***rofessors can live happily ever after (wonders of teaching/tenure): it depends on them how happily they enjoy their ever after (the legends not even retire, ever after). No place is more prestigious and safer than the University!

Looking forward: I suspect I am/will be back once again in my life's inescapable wave of uncertain heterogeneity. I believe this phenomenal training will guide the future; & technology will have/maintain a strong market share of my contributions in future adventures (insert here a complex ***smile!***).

- THE END -

*Middle embodiments like CNRS/Faunhofer-like were not investigated as Industry/Academia!

Appendix

The appendix provides some extended discussions, tables, figures, general document statistics etc. of the thesis. A1 provides a comprehensive survey of referred patents in relation with the patents study discussions of chapter 2.3.1. A2 provides extended tables and discussions for chapter 4.2. A3 provides abstract perspectives for NoC and MRAMs for Heterogeneous MPSoC platforms, and for routing architecture of eFPGA for efficiency and 3rd party hard blocks (IP) integration. A4 provides general statistics of the thesis document.

A1: General overview/survey of FPGA vendors Patents

This appendix provides a comprehensive overview of the patents survey referred in [3] in perspective of the discussions during chapter 2.3.1 for obtaining general broad idea about state of art solutions for research motivations and challenges. Deeper details of them are beyond scope of thesis discussions. The additional contributing part of patents survey reference section [3] and the way it is presented is collecting key fundamental patents which are helpful for FPGA researchers for motivations and provide them an easy access to vast (thousands) patents of vendors for deeper investigations based on their research area. The discussions are categorized in different key areas/aspects of FPGAs research.

It is interesting/important to note that all patents provide scientific ideas with the invention presented in them and are mostly but not necessarily always (particularly for giant vendors) used in the production devices. The prime intent of the patent is to prevent the competitors/new-comers from using that idea in as wide sense as possible.

Foundation Patents

This section briefly outlines the foundation patents of some (others include Lattice, Atmel, QuickLogic etc.) leading FPGA vendors, granted to their founders in late 1980s for providing a historical perspective. The later sections will address in detail particular aspects and advancements areas of present day FPGAs.

[Xilinx_Freeman89, 3.1] described a programmable logic and interconnect architecture. This patent is among the most famous patents in the history of semiconductor industry, it is credited for the invention of FPGA and is the foundation of present day LUT based SRAM FPGAs, which is near universal among almost all FPGA vendors now. [Altera_Hartmann88, 3.35] described a programmable logic device with AND and OR gate arrays using the CMOS EPROM floating gate technology. [Actel_ElGamal89, 3.55] described a programmable interconnect architecture using Anti-fuses.

1- Logic Block

All commercial FPGAs use clustered logic blocks. In addition heterogeneous capabilities e.g. carry chains, multiple functionalities of LUT etc. are also common (chapter 2.1). This section provides a brief survey of logic block patents.

[Xilinx_Kondapalli07; 3.5, 3.6][Xilinx_Young08, 3.9] illustrate in detail the LUT6 based logic block architecture (Slice). The sheer complexity and multitude of heterogeneous functionalities in state of art can be observed in it. The logic block in addition to classical logic implementation of LUT is also optimized for using LUT6 as RAM, shift register etc. Special architectural innovations allow efficient implementations of adders, accumulators, multipliers etc. The highly depopulated feedbacks connectivity to the LUT inputs can also be observed in it.

[Altera_Lewis10, 3.36] and [Altera_Kaptanoglu06, 4.37] provide some insight to the adaptable LUT innovations of Altera (chapter 2.1.2) by creation of logic elements composed of group of smaller LUTs compared to a full LUT6 that are connected in a special manner to provide more silicon efficient solution.

[Altera_Lewis10, 3.38] describes an efficient manner of connecting the logic elements in the logic block (LAB). [Altera_Lewis08, 3.39] presents architecture enhancements of logic block by having more than one flip-flop per LUT leading to more registered outputs than combinational or combinational output that can drive more than one flip-flop. [Altera_Padalia10, 3.40] presents CAD techniques for efficient clustering of logic elements in logic blocks (LAB) in connection with placement information. [Altera Leventis06, 3.41] presents interconnect structure for inside and outside of logic block. The above mentioned patents also provide insight on the challenges and potentials of depopulation of local interconnect for the feedbacks and input signals and efficient connectivity of the combinational and sequential outputs of logic elements

[Actel_Feng09, 3.56] describes an efficient depopulated local interconnect architecture in two levels of multiplexers which are dramatically smaller in size compared to the two levels (CB+LI as we discussed in chapter 4.2) of classical VPR-like explored architectures (this patent is quite interesting for VPR-like architecture researchers). [M2000_Reblewski03, 3.72] presents highly optimized depopulated logic block architecture for feedbacks. It also uses logic block outputs depopulation (some logic elements outputs are only used for feedback hence decreasing total number of logic block outputs for routing network). [M2000_Lepape09, 3.73][M2000_Lepape10, 3.74] provides enhancements to previous mentioned [3.72] to add heterogeneity in the logic block, e.g. adders, special-function elements etc. inside the logic block.

In addition to adding heterogeneity in logic blocks, research is also conducted in heterogeneous use of logic blocks or their architecture to exploit different benefits. [Xilinx_New01, 3.7][Xilinx_Kaviani03, 3.8] present enhancement of using some of the interconnect multiplexers (similar to SB-eLB multiplexers in this thesis work) which route signals to input of logic elements (or LUTs) to be used also for implementing multiplexers for mapped applications (saving logic resources used if implemented using logic blocks). [TierLogic_Madurawe08, 3.93] discusses integration of carry logic inside the LUT hardware. [TierLogic_Dorairaj09, 3.94] presents use of latch to implement logic for enhancing logic capacity.

2- Routing Architecture

Routing is most interesting and complex part of FPGA architecture research. All FPGA vendors have their specialized architecture that differentiates them from their competitors; this is further backed up by the differentiated CAD tools of corresponding vendors which efficiently utilize the hardware for mapped applications. Like we observed sheer complexity of logic block in previous section same goes for routing architecture in state of art. The routing is generally composed of complex hierarchies of heterogeneous length of wires along with specialized electrical implementation (we will discuss in more detail in later section on silicon implementation) innovations for buffers, specialized multiplexers etc. to further enhance capabilities and efficiency of architecture. This section provides a brief survey of architectural patents.

[Xilinx_Tavana99, 3.10] presents in detail the routing architecture of tile based (common in all commercial devices, will discuss further in next section) FPGAs. It also helps providing insight to complex routing structure of state of art FPGAs along with intense depopulation of routing flexibility (what was partly addressed by this thesis work also in chapter 4.2) in the routing matrix (SB) and logic block. The logic block architecture presents specialized form of multilayer switching levels with high depopulation which overall is further silicon efficient compared to classical CB+LI (VPR-like) architecture and complete single layer unified SB architecture that we addressed in chapter 4.2. [Xilinx_Young99; 3.11, 3.12] provides details about hierarchical long lines with bidirectional and unidirectional interconnects. [Xilinx_Young02, 3.13] describes an efficient method of extending the routing architecture of architecture while maintaining the configuration compatibility with original architecture. [Xilinx_Young07; 3.14, 3.15] describe in detail with illustrative diagrams the long and diagonal lines based architecture innovations, the hop patterns, staggered routing tracks etc.

[Altera_Lewis05, 3.43] presents the 3-sided routing architecture (chapter 2.1.2) where the logic block (LAB) connects to three neighboring channels providing attractive features to exploit layout of LAB, block

memories connection ease, better redundancy control for defective LABs if needed etc. It also explains the issues related with 1-sided, 1.5-sided, 2-sided and 4-sided architectures. [Altera_Betz04, 3.44] presents heterogeneous interconnection architecture where some routing tracks are faster than the others and logic block pins are connected to the tracks in a heterogeneous manner to exploit area versus speed scenario in a silicon efficient way. [Altera_Hutton08, 3.45] presents hierarchical organization of logic blocks to efficiently share special resources like clock etc. [Altera_Lewis08, 3.42] presents special interconnect lines to allow connecting logical elements in different (nearby) logic blocks directly. [Altera_Hutton07, 3.47] presents an efficient way of directly transporting the date to/from IO pads to deep inside the core through dedicated long lines at a higher speed.

[Actel_Lien04, 3.59] presents a tile based architecture where each tile is composed of an array of logic blocks, the tiles are surrounded by JTAG, Configuration and BIST interfaces on the outer edge of tiles. The architecture also consists of hard blocks of RAM between outer perimeter of tiles and I/O pads. [Actel_Kaptanoglu09; 3.60, 3.61] present the routing architecture for a specialized semi hierarchical FPGA architecture with array of tiles of large clusters of logic blocks.

3- Column (tiles) based architecture

Using tile based mesh architecture with columns of same tiles (logic blocks, hard blocks etc.) is universal in almost all commercial FPGAs. It allows high layout scalability, efficient design of routing networks, clock distributions, redundancy control (for defective areas) and several other benefits. This section provides a brief overview of patents describing/advocating tile based architectures.

[Xilinx_Young09, 3.16] explains in detail the layout challenges for state of art FPGAs. It presents benefits and issues related to column based tile layouts where each column is entirely composed of same kind of tiles (the logic blocks, hard blocks etc.). By adjusting the width of the column to an effective value efficient layout is achieved. It also presents innovation for putting I/O pads instead of classical boundary of device, directly in the device (core!) in columns of I/O pads which provide superior benefits compared to classical boundary I/Os in different aspects. [Xilinx_Trevor09, 3.18] presents inserting application specific blocks columns in the device (structured-asic-like/asic-like scenario) to create application specific devices without disturbing the fundamental architecture. [Xilinx_Tavana99, 3.10] also presents tile based architecture for layout scalability by repetitive use of layout of same tiles (is the foundation motivation of column-based/tile-based architectures in commercial FPGAs). [Altera_Lewis05, 3.43] (That we discussed in previous section also) presents the routing architecture for similar tile-based columnar FPGA architectures motivations and exploitations for routing and hard macros etc. [Actel_Feng08, 3.58] presents a tile-able architecture with logic blocks distributed in large groups of tiles (semi hierarchical architecture topology), the hard blocks are integrated in a specialized manner outside the core fine grained logic blocks area, however the logic blocks use highly regular architecture (keeping tile based motivation for layout).

4- Hard Blocks

The architectural heterogeneity (different kind of blocks in addition to heterogeneity of functionalities of fundamental logic block which we discussed above) is common in all commercial FPGAs with strong following of column based architecture motivations as discussed above. This section provides a brief overview of patents in this regard.

The hard blocks exist in different forms and varieties in commercial FPGAs. The general purpose blocks like small memories, multipliers etc. are common in commercial FPGAs and follow the column based approach motivations. Second levels of hard blocks (some are becoming general also) include components like clock managers, high speed serial I/Os etc. they are also generally deployed in column based fashion. [Xilinx_Young09, 3.16] that we discussed above for column based architectures present aspects of these blocks along with basic ideas of third level of further complex blocks like e.g. a processor core. To achieve

greater level of flexibility and efficient utilization of hard blocks it is often desired/needed that the hard blocks provide multiple options and are efficient to be fractured down or combined together to form bigger blocks, e.g. the general purpose blocks of memories (SRAM) are designed to have multiple and adaptable capabilities. [Xilinx_Young99, 3.19][Xilinx_Pang02, 3.20][Altera_Cliff07, 3.48] present several aspects of/for memory blocks in FPGAs. [Xilinx_New02, 3.27] presents methods for multiplier hard block integration. [Xilinx_Douglas04, 3.28] presents interconnect architecture for connectivity of hard blocks like microprocessors.

Another style of putting hard blocks is separating them from the main fine grain logic blocks portion and make the FPGA architecture in a relatively SoC-like fashion where different blocks are combined together (large block of dense layout-friendly logic blocks and different kinds of hard blocks including analog components). Such style is prominent for FPGA devices of Actel. [Actel_Beal10, 3.65] presents one such example. [Actel_Bakker09_3.64] presents a programmable system on chip architecture with block of programmable logic along with digital, analog blocks, memories and microprocessor (similar to smart fusion devices of Actel which we observed in chapter 2.3.4).

5- Silicon implementation (electrical design differentiations)

All commercial FPGAs are custom designed with huge layout efforts to create optimal electrical implementation of the logical architecture at the target node. Throughout the patents been addressed above, most of them provide some aspects of physical design challenges. While creation of layout for the logical design of FPGA, several electrical implementations are possible. This aspect alone is huge research area and source of a competitive edge among the vendors. From almost smallest base components (e.g. LUT, Flip-Flop, SRAM cells) to global routing architecture and hard blocks; numerous exploitations are done to address several issues e.g. area efficiency, speed, power etc. This section provides a brief overview of some of the aspects in connection with the discussions in chapter 2.1.2 and chapter 4.

[Xilinx_Young06, 3.2] presents LUT design using different kind of transistors (varied in gate oxide thickness, operating voltage, gate length etc.) for configuration, stages of multiplexer design, buffers etc. to create a silicon efficient power versus performance solution. [Xilinx_Young06, 3.3][Xilinx_Young06, 3.4] present architecture structures for creating LUT-6. [Xilinx_Duong97, 3.25] presents the single buffered six pass transistor based switch box element. [Altera_Lewis06, 3.50] presents low power routing multiplexers with efficient silicon implementation to reduce static and dynamic power, different modes of operation (power vs speed) etc. [Altera_Lewis07, 3.51] describes merged logic elements routing multiplexer structure for improved speed of transfer of data between logic elements in two different logic blocks. [Altera_Lewis09, 3.49] presents the methods for controlling speed and power for the FPGA (is/similar-to the programmable power technology of Altera).

6- Configuration

The configuration architecture provides the configuration infrastructure of FPGA. There can be multiple types of configuration architectures for a same logical architecture of FPGA. The configuration architecture/methods/systems are a standalone diverse research/differentiation area among FPGA vendors with multiple levels and dimensions of challenges like configuration memory type (volatile/non-volatile), advanced features (partial/dynamic reconfiguration etc.), security (particularly for non-volatile FPGAs), radiation hardness etc. This section provides a brief overview of patents on configuration for general overview (configuration was not main objective/concern of this thesis work and was not deeply investigated like logic blocks, routing architectures etc.).

[Xilinx_Rau02, 3.21] details a configuration memory architecture infrastructure and programming methodologies for SRAM-based FPGA. [Xilinx_Schultz01, 3.22] addresses in detail methods and structures for configuration, particularly advanced features like partial reconfiguration, the feeling of

complexity of configuration infrastructure in state of art can be observed in it. [Xilinx_Trimberger07, 3.23] presents different methods for configuration. [Xilinx_Trimberger07, 3.24] presents methods and infrastructure for overriding defective configuration memory in FPGA. Security is a major concern for FPGAs (particularly non-volatile configuration based, where configuration has to be loaded from external source at each power up) to prevent illegal use of design by stealing the configuration bitstream. [Xilinx_Pang05, 3.26] addresses methods to prevent reading back the decrypted bitstream out of the device which was loaded in encrypted from outside memory/system (decryption done by FPGA internally). [Actel_McCollum09, 3.67] present the structure of non-volatile memory cell in the CMOS inverter form with one of transistor non-volatile (Flash etc.). [Actel_Plants08, 3.66] presents hardware structures for controlling SRAM cells by non-volatile Flash cells. [Actel_Dhaoui10, 3.68] presents radiation-tolerant Flash-based memory cells architectures (the silicon overhead of added transistors to make the system radiation hardened can be observed in it).

7- Beyond Classic/Un-conventional/New Trends

Throughout our discussions above we addressed the areas and challenges which are common and fundamental in almost all FPGAs and still are the largest research area. However over the past few years, researchers in both industry and academics have also started investigating in un-conventional dimensions to address challenges of FPGAs, to further strengthen their technological and commercial position in changing landscapes of industry. This section provides a brief overview of patents (vendor by vendor basis) of relatively un-conventional architecture innovations, trends etc.

[Xilinx_Trimberger07, 3.29] presents a *time-multiplexed interconnect* infrastructure composed of two layers of routing, one conventional circuit switching routing network of FPGAs and a secondary shared subway routing network which can be time-multiplexed. The classical network can be used for critical signals and lower speed signals can be transmitted through the time-multiplexed routing network. The invention in a wide sense provides a *combined circuit switching plus packet switching routing network*. [Xilinx_Young09, 3.30][Xilinx_Young09, 3.31][Xilinx_Young10, 3.32] present architectures for implementing *hard handshake logic* in the routing architecture. The inventions are presented in the scenario of clock skew challenges in large FPGAs for meeting setup and hold time requirements of synchronous design implementations. [Xilinx_Rahman09, 3.33] and [Xilinx_Chaware11, 3.34] address motivations and innovations of *3D die-stacking* of multiple FPGAs.

[M2000_Reblewski09, 3.75] presents a selective *packet-switching* based routing infrastructure to communicate between different clusters of logic. [M2000_Reblewski10, 3.76] presents methodologies for addressing the issues of yield during fabrication for fabricated devices by assigning functionality of a defective logic portion to other portion managed through hardware. The above mentioned inventions are presented in the scenario of yield issues for fabricating large FPGAs; use of packet-based network facilitates identification and remapping of defective clusters of logic to other available clusters after production. [M2000_Ebeling10, 3.77] presents the invention of use of *MEMS* (Micro-Electro-Mechanical Systems) for FPGAs switching elements for crossbars, LUTs etc. MEMS can allow exploiting some special features like, low resistance, zero-leakage, non-volatility etc. compared to classical transistor based configuration system.

[Achronix_Manohar09, 3.79][Achronix_Manohar10, 3.80][Achronix_Manohar09, 3.78] present inventions for automatic conversion of synchronous design into asynchronous designs which can be mapped on an *asynchronous hardware*. The industry historically has and still is synchronous designs dominated; the tools and flows are well developed for synchronous designs, furthermore due to that dominance a vast amount of existing designs are synchronous. The challenge for automatic and seamless conversion of synchronous designs into asynchronous designs taking benefits of asynchronous designs and efficiently implementing them on asynchronous hardware is the background motivation of these inventions (differentiation of

Achronix FPGAs). [Achronix_Manohar10, 3.81] presents in detail the hardware components of the asynchronous FPGA and methods to implement asynchronous applications on them. [Achronix_Manohar10, 3.82] presents challenge for fault tolerance in asynchronous circuits; due to absence of clock, the transient faults can cause glitches that cannot be filtered and can disrupt the computation. It presents the invention for fault tolerant asynchronous circuits.

[Tabula_Redgrave09, 3.83] describes the invention of *packet-switching* use for fast configuration and debug of the programmable device along with the hardware details. [Tabula_Schmit11, 3.84] describes methods and underlying hardware to implement the standard input design in multiple cycles (sub-cycles) on the time multiplexed device which reconfigures rapidly and efficiently in an innovative manner compared to classical equivalents (dynamic/partial reconfiguration). [Tabula_Hutchings08, 3.85] presents programmable device with *hybrid circuits* that can function for both logic and interconnect implementation, the presented novel architecture exploits the use of multiplexers which in addition to be controlled by configuration data can also be controlled by signals internally computed (mapped application). [Tabula_Teig09, 3.86] describes the invention of a programmable device with storage elements in the routing fabric (in wide sense storage at output of routing multiplexer) which can be used for different un-conventional techniques like e.g. configuration cell (adaptive unlike fixed configuration cells in classical FPGA architectures). [Tabula_Rohe10, 3.87] presents novel ideas of long wires in routing network with heterogeneous patterns among different logic blocks (classically the routing network is homogeneous for all blocks) which are repeated in a specialized regular fashion thereby increasing the interconnectivity among logic blocks by providing more diversity and avoiding redundant routing behavior of classical FPGA routing architectures.

[TierLogic_Madurawe09, 3.88] addresses three dimensional design of programmable device. The invention is discussed in the perspective of easy FPGA to ASIC design migration. The configuration memory of FPGAs occupies significant silicon area. The invention presents concept of 3D stacking of configuration memory on the logic using technologies like TFT (Thin Film Transistors). [TierLogic_Madurawe10, 3.89] and [TierLogic_Madurawe10, 3.90] further illustrate the concept FPGA to ASIC migration in perspective of 3D stacked configuration memory. [TierLogic_Madurawe91, 3.91] presents time-multiplexed routing interconnect in perspective of 3D stacked configuration memory.

A2: Extended tables and discussions of Chapter 4

This appendix provides the extended tables of chapter 4which present the detailed experiment results of some of the discussions which referred to this appendix for further details. A comprehensive summary of these extended tables to make the cross interlink with chapter 4 is presented below.

SB-R explorations (section 4.2.2)

Table A2.1 provides the silicon statistics for the seven SB-R exploration tiles for channel width 36 which were used in the second iteration (corresponding to table 4.7 for channel size 48) for proof of concept.

Motivation and proof of Chan max. used vs Chan min. needed in published works (section 4.2.2)

Channel size is one of the most important and challenging aspect of FPGA architecture research. Finding the minimum channel width needed to effectively route an application is an important part of architectural exploration and decision making (as we discussed in detail in chapter 4.1.4). [Betz&Rose99, 4.1] proposed a binary search method to repeatedly iterate PAR to reach minimum channel width. While this is one of the best ways to actually know the minimum channel needed but was quite complicated for our eFPGA Creator flow in current form (chapter 3) as unlike VPR which automatically builds new architecture for iteration eFPGA Programmer needs to be provided with new architecture for each iteration created by eFPGA Creator which indeed is lengthy (creation + silicon implementation). However knowledge of minimum channel is obligatory for exploration, otherwise the detailed explorations that we conducted (with channel width set to 48) could easily lead to false conclusions about the architectural innovations we explored, since if channel width is arbitrarily at a very high size compared to benchmarks need, making successive iterations for tuned customization (as we did) and observing successful PAR could be due to inherent flexibility of vast routing available due to large channel size than needed giving router enough freedom to route despite reduced architectural routing flexibility due to customization. To address this issue we took the benefit of eFPGA Creator analysis tools suite (chapter 3) which directly inform the maximum of the channel *used* (in connection with minimum of channel *needed*, which is common in VPR-like research explorations). This assumption was theoretically also sound since the fact than Placement and Routing are independent in eFPGA Programmer like VPR (simulated annealing) so Placer is unaware of the routing architecture and performs the best placement which the router finally routes (not always the case in state of art as we discussed in chapter 2). Table A2.2 practically illustrates that this was a good assumption, it shows the fundamental base architecture (highest flexibility) successively iterated for decreasing channel width (creating VPR like scenario) to find minimum overall channel width. It can be seen there is negligible effect when routing constraints are tightened for router by decreasing physical channel size and ultimately benchmarks start failing to be routed. Hence the maximum channel used parameter of our experiments provides a good approximate equivalent of minimum channel needed, while saving enormous time for repeated iterations, allowing focusing on main investigations of customizations.

Combined SB-R + SB-eLB optimizations (section 4.2.4)

Table A2.3 provides detailed silicon statistics for best cases SB-R + SB-eLB explorations for theme 2 and theme 3 tiles. Table A2.4 and A2.5 provide detailed PAR statistics of benchmark applications mapping on theme 2 and theme 3 tiles based architectures (theme 1 failed to map benchmarks, so is not shown).

		Base	Theme 1			Theme 2		
ST 65LPSVT _1.3V-m40C		tsr_4436_F	tsr_4436_noN	tsr_4436_noNE	tsr_4436_50pAll	tsr_4436_c1	tsr_4436_c2	tsr_4436_c3
Conf. bits	Total	396	378	360	360	324	324	324
	SBOX	328	310	292	292	256	256	256
	eLB	68	68	68	68	68	68	68
Area (um²)	Total	12077	11596	11139	11139	10312	10200	10200
	SB-R	1819	1544	1270	1270	904	878	878
	SB-eLB	5232	5232	5232	5232	5232	5191	5191
	eLB	418	418	418	418	418	368	368
	Conf.	3419	3261	3124	3124	2796	2796	2796
Logic Density (LUT4/mm²)		331	345	359	359	388	392	392
Area Normalized to F		1.00	0.96	0.92	0.92	0.85	0.84	0.84
Power	Static (nW)	371	357	344	344	318	307	307
	Dyn*. (uW/MHz)	3.2	3.02	2.83	2.82	2.51	2.44	2.44

Table A2.1: Silicon statistics of SB-R experimented tiles at LUT4, Cl4, Ch36 (*pessimistic)

t_F-Lut4_Cl4_ChX	For Ch-48	For Ch-36	For Ch-24
Circuits	PAR Ch. Max	PAR Ch. Max	PAR Ch. Max
ex5p	20	21	20
tseng	20	21	20
apex4	35	33	RF
misex3	25	25	21
alu4	29	26	23
diffeq	21	22	21
seq	33	32	RF
apex2	36	30	RF
k2	30	29	24
s298	36	32	RF
ex1010	35	33	RF

Table A2.2: PAR statistics of base tile (_F topology) for decreasing channel width

		Base	Theme 3		Theme 2	
ST 65LPSVT _1.3V-m40C		t_4448_F	t_4448_rc2-enoNE	t_4448_rc3-enoNE	t_4448_rnoNE-ec1	t_4448_rnoNE-ec2
Conf. bits	Total	468	356	356	388	388
	SBOX	400	288	288	320	320
	eLB	68	68	68	68	68
Area (um²)	Total	15023	9448	9448	8687	8687
	SB-R	2440	1198	1198	1673	1673
	SB-eLB	6720	3603	3603	1989	1989
	eLB	418	418	418	418	418
	Conf.	4043	3124	3124	3403	3403
	Buff.	1402	1105	1105	1204	1204
Logic Density (LUT4/mm²)		266	423	423	460	460
Area Normalized to F		1.00	0.63	0.63	0.58	0.58
Power	Static (nW)	452	296	296	275	275
	Dyn.* (uW/MHz)	3.67	2.26	2.26	2.08	2.08

Table A2.3: Silicon statistics of best case (SB-R + SB-eLB) hybrid tiles (*pessimistic)

Circuit	t_4448_SB-RnoNE_SB-ec1				t_4448_SB-RnoNE_SB-ec2			
	Ch. Max	Ch. Avg	R. Iter	Delay(ns)	Ch. Max	Ch. Avg	R. Iter	Delay(ns)
ex5p	32	7.6	6	11.04	32	7.2	6	9.64
tseng	27	10.8	6	11.1	24	10.8	5	10.85
apex4	43	25.4	40	17.2	44	25.76	93	17.35
misex3	33	12.9	6	10.1	33	13	7	10.66
alu4	37	18	8	13.4	39	18	9	12.24
diffeq	30	11.5	6	13	29	11.4	6	13
seq	42	25.2	45	12.5	41	25.48	89	12.78
apex2	41	24	16	13.75	45	24	20	13.77
k2	40	17.6	10	12.06	38	17.5	8	11.14
s298	RF				RF			
ex1010	RF				RF			
A. Mean	**36**	**17**	**15.9**	**12.68**	**36**	**17**	**27**	**12.38**

Table A2.4: benchmarks PAR statistics for SB-R+SB-eLB theme 2 tiles

Circuit	t_4448_SB-Rc2_SB-enoNE				t_4448_SB-Rc3_SB-enoNE			
	Ch. Max	Ch. Avg	R. Iter	Delay(ns)	Ch. Max	Ch. Avg	R. Iter	Delay(ns)
ex5p	26	5.6	6	8.98	26	5.8	7	8.6
tseng	26	10.6	7	11.1	26	10.5	6	10.5
apex4	38	19.4	10	14.86	41	19.7	14	14.26
misex3	32	10.5	7	9.38	30	10.2	7	9.42
alu4	31	14.3	6	12.04	32	14.27	7	11.29
diffeq	29	11.06	7	12.15	28	11	6	12.2
seq	37	19.1	8	11.75	38	19.4	8	11.19
apex2	37	18	7	13.32	38	18	8	12.5
k2	35	14.1	7	11.24	36	14.3	7	11.36
s298	37	20.8	13	18.59	41	21	13	18.8
ex1010	RF				RF			
A. Mean	**32.8**	**14.34**	**7.8**	**12.34**	**33.6**	**14.4**	**8.3**	**12.01**

Table A2.5: benchmarks PAR statistics for SB-R+SB-eLB theme 3 tiles

Revisit to LUT size effect on silicon efficiency (sections 4.1.2, 4.2.5)

In continuation and relation of section 4.1.2 (table 4.4) and perspectives discussions of section 4.2.5, experiments were conducted for multiple LUT sizes with base architecture (_F topology) in the light of silicon tradeoffs, mapping efficiency and benchmarks mapping. Table A2.6 shows the silicon statistics of base tiles for different LUT sizes (with cluster 4 and channel width 48). Tables A2.7 and A2.8 show detailed results of mapping benchmark applications on corresponding architectures. It can be seen that channel size demands are almost similar for all cases (as we observed and discussed in section 4.1.2), however what is interesting to observe is that LUT6 based architecture is relatively slower than LUT4 (the theoretical benefits of LUT6 is degraded due to poor mapping efficiency). TableA2.9 and corresponding figure A2.1 illustrates the findings relative to LUT4. We can see that LUT3 clearly is poor for silicon efficiency. LUT5 provides an interesting value proposition (better than LUT4 for explored benchmarks). The LUT6 silicon tradeoff due to poor mapping efficiency is visible in graph along with slower speed, which results from the fact that individually LUT6 has higher delay than smaller LUTs and since the total LUTs needed for LUT6 (mapping efficiency) do not shrink enough for PAR compared to LUT4 e.g.

ST 65LPSVT _1.3V-m40C		t_4448_F (LUT4)	t_3448F(LUT3)	t_5448F(LUT5)	t_6448F(LUT6)
Conf. bits	Total	468	408	560	716
	SBOX	400	372	428	456
	eLB	68	36	132	260
Area (um^2)	Total	15023	12464	18081	22793
	SB-R	2440	2396	2396	2396
	SB-eLB	6720	5040	8400	10080
	eLB	418	216	818	1625
	Conf.	4043	3540	4759	6080
	Buff.	1402	1272	1708	2612
Logic Density (LUT/mm^2)		266	321	221	175
Power	Static (nW)	452	371	552	717
	Dyn*. (uW/MHz)	3.67	2.99	4.63	6.42

Table A2.6: Silicon statistics of base tile for different LUT sizes (*pessimistic)

Circuit	t_6448_F (LUT6 arch.)					t_4448_F (LUT4 arch.)				
	No. LUT6	Ch. Max	Ch. Avg	R. Iter	Delay(ns)	No. LUT4	Ch. Max	Ch. Avg	R. Iter	Delay(ns)
ex5p	154	16	2.9	4	10.05	240	20	4.27	5	11.77
tseng	643	27	8	6	15.68	743	20	8.4	5	13.6
apex4	548	36	14.9	7	23.9	742	35	16.4	6	20.99
misex3	231	20	5	5	13.5	426	25	8.26	5	13.1
alu4	410	25	7.8	5	18.7	614	29	11.49	6	16.41
diffeq	566	26	9.45	6	21.01	714	21	8.93	5	15.64
seq	555	39	13.23	7	23.19	781	33	15.37	6	16.7
apex2	476	32	11.6	7	20.5	714	36	14.2	7	18.38
k2	438	36	11.1	7	18.96	546	30	11.1	5	17.28
s298	585	33	15.4	7	26.8	856	36	17.3	7	23.87
ex1010	632	41	18	7	26.15	922	35	18.7	8	20.45
A. Mean	**476.2**	**30.1**	**10.7**	**6.2**	**19.86**	**663.5**	**29.1**	**12.22**	**5.91**	**17.11**

Table A2.7: benchmarks PAR statistics for base tile architecture (LUT6, LUT4)

Circuit	t_3448_F (LUT3 arch.)					t_5448_F (LUT5 arch.)				
	No. LUT3	Ch. Max	Ch. Avg	R. Iter	Delay(ns)	No. LUT5	Ch. Max	Ch. Avg	R. Iter	Delay(ns)
ex5p	354	22	4.36	6	13.6	193	22	4	5	9.75
tseng	979	21	6.8	5	16.2	671	26	7.7	5	11.97
apex4	964	33	14.8	6	24.1	636	37	16	7	18.05
misex3	606	29	8	6	16.9	327	22	6.4	5	11.5
alu4	867	28	11	6	20.93	481	27	9.8	6	13.32
diffeq	1031	21	8.2	5	19.85	598	27	9.5	6	15.44
seq	1021	34	13.6	7	23.94	691	35	16	7	15.4
apex2	891	27	11.34	7	20.39	621	30	13.3	7	15.41
k2	713	32	9.7	6	19.8	527	35	12.3	6	15.9
s298	1126	33	15	7	34.31	709	36	17.3	7	19.42
ex1010	1126	32	16	7	21.75	725	35	16.9	7	18.9
A. Mean	**879.8**	**28.4**	**10.8**	**6.2**	**21.1**	**561.73**	**30.2**	**11.7**	**6.2**	**15**

Table A2.8: benchmarks PAR statistics for base tile architecture (LUT3, LUT5)

ST65LPSVT_1.3V-m40C	t_3448F(LUT3)	t_4448_F (LUT4)	t_5448F(LUT5)	t_6448F(LUT6)
Logic Density (LD)	321	266	221	175
LUTs needed	880	664	562	476
LD/LUT needed	0.365	0.401	0.393	0.368
Avg. BM Delay (ns)	21.1	17.11	15	19.86

Table A2.9: Relative comparison of different LUT sizes

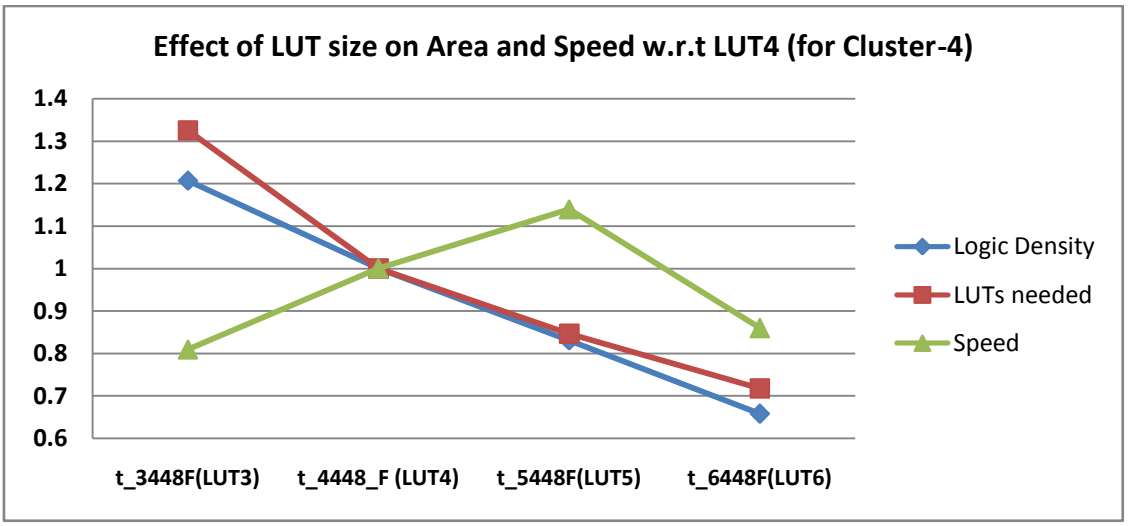


Fig. A2.1: Relative comparison of different LUT sizes with respect to LUT4

Power vs Speed: effect of threshold voltage and process type (section 4.2.4)

Table A2.10 shows the detailed benchmarks mapping results (variation in speed) on same logical architecture (area optimized golden tile) that is implemented on different process options of 65nm. The effect of changing threshold voltage on speed and significant change in speed from LP (Low Power) to GP (General Purpose) is clearly visible which comes at price of power consumption (addressed in section 4.2.4).

t_4448_rnoNE-ec1		Critical Path Delay (ns), ST65nm process					
Circuit	**Ch. Max**	**LPLVT**	**LPSVT**	**LPHVT**	**GPLVT**	**GPSVT**	**GPHVT**
ex5p	32	14.42	18.11	25.97	8.21	10.98	15.32
tseng	27	14.5	18.16	26.75	8.35	10.95	15.05
apex4	43	22.33	28.72	40.98	12.97	17.24	23.8
misex3	33	12.99	16.69	23.67	7.68	10.04	13.73
alu4	37	17.22	21.98	30.6	9.97	13.31	18.2
diffeq	30	16.7	21.44	31.1	9.54	12.55	17.9
seq	42	16.08	20.75	29.4	9.37	12.3	17.47
apex2	41	18	22.68	33.1	10.11	13.4	19.08
k2	40	15.68	20.01	28.44	9.04	11.84	16.41
A. Mean	**36**	**16.44**	**20.95**	**30**	**9.47**	**12.51**	**17.44**

Table A2.10: benchmarks PAR statistics of best SB-R+SB-eLB architecture for varied 65nm process types

A3: NoC+MRAM Perspectives for eFPGAs/eFPGAs-based-Systems

This appendix provides an abstract overview of perspectives of NoC (Network on Chip) and MRAMs for eFPGAs, SoC with eFPGAs and also potentials of unconventional eFPGAs for SoCs. The discussions are highly abstract with no physical hardware proposition/results. They are based on the knowledge gained from the thesis work (industrial survey, experiments done, challenges observed etc.). They can be interesting potential research areas for researchers in general and a long term extension for this thesis work for conventional plus beyond classics for the two foundation axes B (eFPGA architecture) and G (eFPGA in systems) respectively which are briefly outlined with theme diagrams below.

eFPGA/FPGA routing, 3rd party IP integration

Routing is the most challenging and expensive part of FPGA architecture. Chapter 4 addressed in dental the challenges regarding routing. One of particular issue noted was strong variation of channel size requirements from application to application and partial packing of channels even for single application (few high congestion regions/channels advocate channel demands for entire device). Chapter 2 addressed in detail the advancements and trends in industry with growing potentials of programmable technologies. It was also observed that there is a growing interest among FPGA vendors and also academia to investigate beyond classic methods/styles to improve potentials of FPGAs. Appendix A1 outlined several patents demonstrating un-conventional trends that are gaining attention in FPGA vendors for architectural innovations in different dimensions. The growing interest in direct/partial form of packet switching was notable among leading FPGA vendors and some startups. The configuration is another prominent aspect and challenge of FPGAs. Chapter 5.4 addressed the perspectives of emerging MRAMs for FPGAs in variety of new opportunities to exploit/experiment. Figure A3.1 presents an abstract theme diagram of heterogeneous eFPGA (see chapter 3.1.2 for component details) suggesting/advocating the investigation perspectives for NoC+MRAM for enhancing potentials of un-conventional eFPGAs with more focus on reconfigurable data computation element for SoC compared to classical prototype element (which has its own significance but is not always a good/efficient choice).

Heterogeneous Reconfigurable MPSoCs

eFPGAs are symbol of heterogeneity in a system due to programmable hardware. Figure A3.2 illustrates a theme diagram of heterogeneous MPSoC platform. The platform is shown as composition of three main domains. The general purpose domain (GPD) consists of main central processor (single/multicore) like those of ARM/MIPS/Intel etc. large enough to execute complex applications and operating systems. The corresponding application specific domain (ASD) provides series of target dependent IPs as hardwired accelerators. Conventional eFPGAs (Ec) like the ones explored in this thesis work can provide flexibility benefits (product differentiation etc.). The third domain represented as programmable hybrid domain (PHD) is an interesting futuristic research frontier demonstrating the inevitable benefits and potentials of classical and beyond classical eFPGAs for many core processor research frontiers. The PHD is abstractly shown to be composed of different combination of processing elements with packet based intercommunication (NoC). Small RISC machines (Px) with memory (distributed memory scenario). Stand alone beyond classics eFPGAs (Ex), and hybrids PEx elements (RISC with eFPGA co-processor as was briefly outlined in chapter 5 discussions for MRAMs perspectives). It can clearly be observed that eFPGAs have significant potentials of research community to unlock doors for new experiments (they can't be achieved easily by prototyping on standard FPGAs!).

It is interesting to note that these apparently quite forward looking concepts presented in this appendix are not in far future from reality in wide sense. Modern FPGAs have already reached a form quite close to figure A3.2 (GPD + ASD, PHD can be partly prototyped). The Heterogeneous MPSoC platforms are

already composed of GPD + ASD (no eFPGA at present) and several players in industry are trying to find niche for PHD-like IPs/devices in different forms (mostly MPPAs etc. as was observed in chapter 2). eFPGAs are indeed a missing ingredient for Reconfigurable Computing research community also and must be addressed by research community for the advancement of application specific eFPGAs for PHDs exploration, exploitation of emerging memories (MRAMs etc) for eFPGAs and Processors. Research on such dimensions has good potential for ROI for academia and industry for future (at least 5 years down the road when such technologies will hopefully start getting mature).

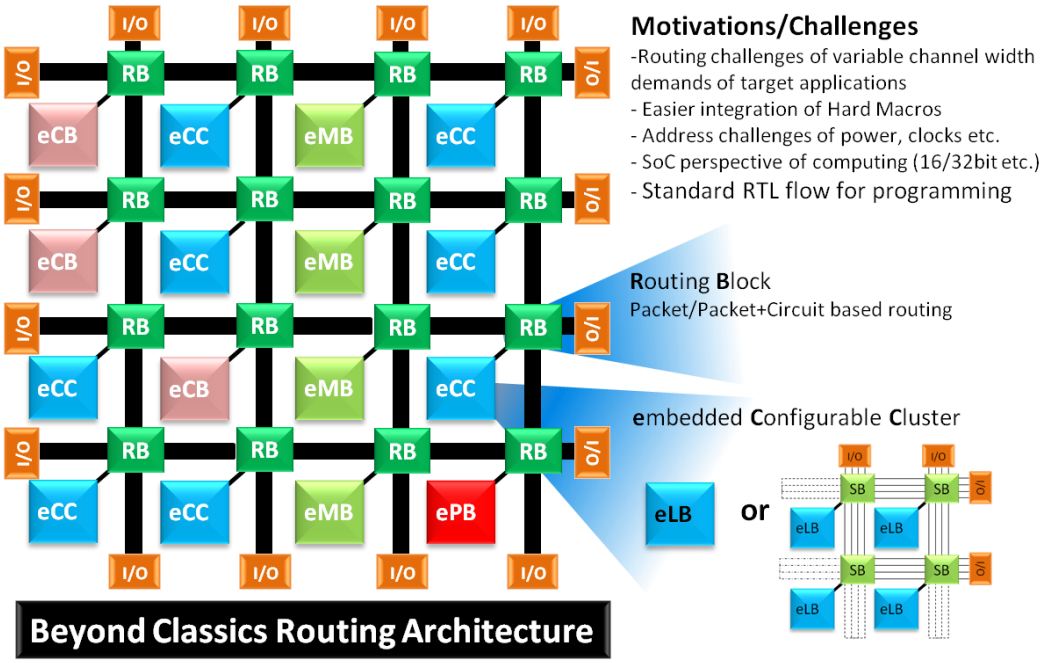


Fig. A3.1: Beyond Classics Routing Architecture potentials in/for eFPGA

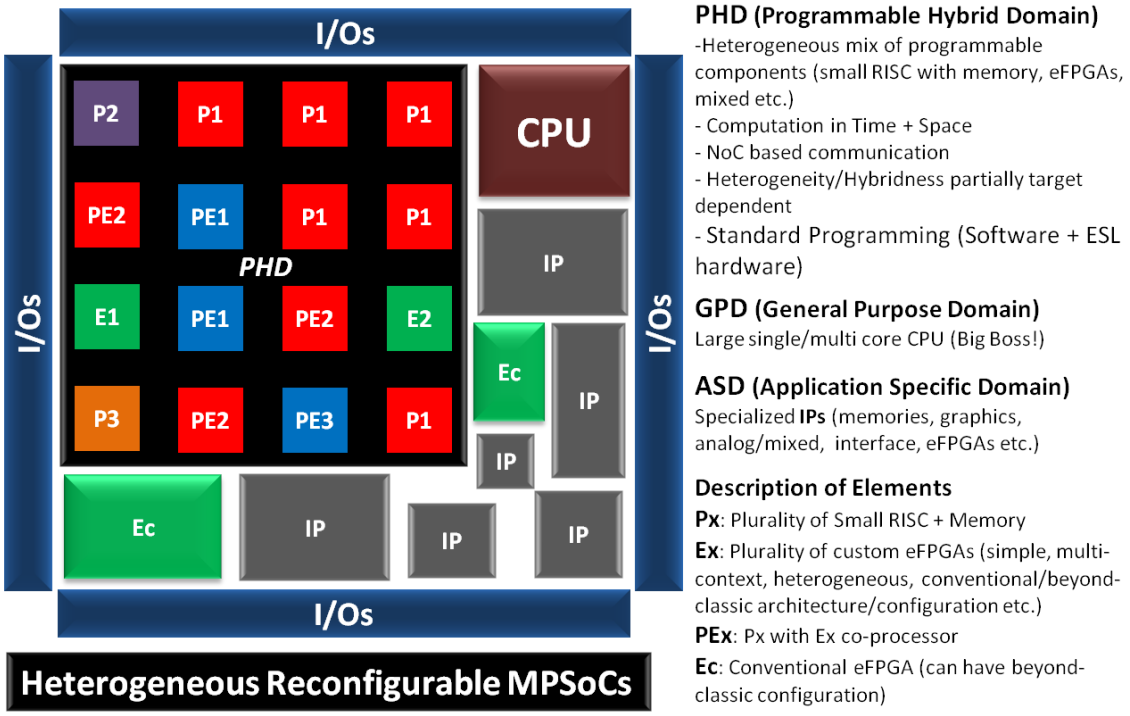


Fig. A3.2: Heterogeneous Reconfigurable MPSoCs (SoC with PHDs)

A4: Document Statistics

This appendix provides approximate statistics of thesis report for general information/overview.

The thesis report is written with Microsoft Office 2007, in Times New Roman 10.5 font with 1.15 Line Spacing on A4 paper with normal border of one inch on all four sides.

Table A4.1 provides approximate statistics for all the six chapters (Core), the front-end (everything before chapter 1), the back-end (everything after chapter 6 excluding References) and References. The approximate statistics are based on Microsoft Word statistics.

Table A4.2 provides statistics of References section.

		No. Pages	No. Figs.+Tables	No. Words	Words Density (Words/Page)	% of Total (Words)
Core	**Ch-1**	5	1	2391	478	3.0
	Ch-2	44	43	18268	415	23.1
	Ch-3	22	23	7828	356	9.9
	Ch-4	40	53	16513	413	20.9
	Ch-5	24	27	10510	438	13.3
	Ch-6	5	1	2638	528	3.3
	Core Total	**140**	**148**	**58148**	**415**	**73.4**
Others	**Front-end**	13	0	7157	421	9.0
	Back-end	18	13	7517	418	9.5
	References	16	0	6349	397	8.0
	Others Total	**47**	**13**	**21023**	**447**	**26.6**
Thesis draft	**Grand Total**	**191**	**161**	**79171**	**415**	**100.0**

Table A4.1: Approximate Statistics of Thesis report

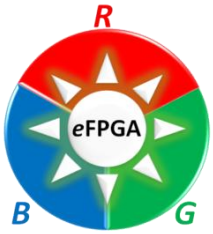
	No. References	% of Total
FPGA Vendors	40	11.8
Industry (General)	51	15.0
FPGAs Patents	94	27.7
Academics	108	31.9
Books and Others	37	10.9
Self	9	2.7
Total	**339**	**100.0**

Table A4.2: Statistics of References

Abbreviations

BLE	Basic Logic Element
CAD	Computer-Aided Design
CB	Connection Block/Box
CMOS	Complementary Metal Oxide Semiconductor
eFPGA	embedded Field Programmable Gate Array
ESL	Electronic System Level (design at higher abstraction level)
FPGA	Field Programmable Gate Array
GP	General Purpose (process type)
GUI	Graphical User Interface
HDL	Hardware Description Language
HM	Hard Macro
HW/SW	Hardware/Software (in HW/SW co-design scenario)
LD	Logic Density (number of LUT per square millimeter)
LI	Local Interconnect
LP	Low Power (process type)
LUT	Look-Up Table
LVT	Low Voltage Threshold (process type, also corresponding High: HVT, Standard: SVT)
MPSoC	Multi Processor System on Chip
MRAM	Magneto-resistive Random Access Memory
NRE	Non Recurring Engineering
NVM	Non-Volatile Memory
PAR	Place and Route
RC	Reconfigurable Computing
ROI	Return On Investment
RTL	Register Transfer Logic
SB	Switch Block/Box
SoC	System on Chip/System on a Chip

PhD Contributions



Major Technical and Commercial contributions of PhD.

Panels

- SoC Configurability- Balancing Manufacturing and R/D Costs. With Gartner, ARM, ST, MIPS, Transmeta and Menta. **IP08**, December 2008, Grenoble, FRANCE [Slides]
- IP/SoC Prototyping. With Bull, ST, Mentor Graphics, EVE, Synplicity and Menta. **IP07**, December 2007, Grenoble, FRANCE [Slides]

International Conferences

- PC member **ReConFig10** (Track: High Performance Reconfigurable Computing), Cancun, MEXICO
- PC member **ReConFig09** (Track: High Performance Reconfigurable Computing), Cancun, MEXICO
- Session Chairman **IP08** (Session: Prototyping 2), Grenoble, FRANCE

International Publications

- *"Survey of new trends in Industry for Programmable hardware: FPGAs, MPPAs, MPSoCs, Structured Asics, eFPGAs and new wave of innovation in FPGAs"*. Syed Zahid Ahmed, Gilles Sassatelli, Lionel Torres, Laurent Rouge. **FPL-10**, 31Aug-2Sep 2010, Milan, ITALY. [Paper][SLIDES*].
- *"A Dynamic Reconfigurable MRAM based FPGA"*. Lionel Torres, Yoann Guillemenet, Syed Zahid Ahmed. **ERSA-10 invited keynote paper**, July 2010, Las Vegas, USA. [Paper]
- *"MRAM based eFPGAs: programming and silicon flows, exploration environments, MRAM current state in Industry and its unique potentials for FPGAs"*. Yoann Guillemenet, Syed Zahid Ahmed, Lionel Torres, Alexandre Martheley, Julien Eydoux, Jean-Baptiste Cuelle, Laurent Rouge, Gilles Sassatelli. **ReConFig09**, December 2009, Cancun, MEXICO. [Paper] [Slides]
- *"FPGA Designer GUI Tools Suite: A complete hardware and software infrastructure for creating customizable eFPGA IP blocks of Menta"*. Syed Zahid Ahmed, Alexandre Martheley, Laurent Rouge, Julien Eydoux, Jean-Baptiste Cuelle. **IPESC09**, December 2009, Grenoble, FRANCE. [Paper] [Slides]
- *"Exploration of power reduction and performance enhancement in LEON3 processor with ESL reprogrammable eFPGA in processor pipeline and as a co-processor"*. Syed Zahid Ahmed, Julien Eydoux, Laurent Rouge, Jean-Baptiste Cuelle, Gilles Sassatelli, Lionel Torres. **DATE-2009**, April 2009, Nice, FRANCE. [Paper] [Slides]
- *"Power consumption reduction explorations in processors by enhancing performance using small ESL reprogrammable eFPGAs"*. Syed Zahid Ahmed, Julien Eydoux, Michael Fernandez, Laurent Rouge, Gilles Sassatelli, Lionel Torres. **ReConFig08**, December 2008, Cancun, MEXICO. [Paper] [Slides]
- *"eFPGA architecture explorations: CAD and Silicon analysis of beyond 90nm technologies to investigate new dimensions of future innovations"* . Syed Zahid Ahmed, Michael Fernandez, Gilles Sassatelli, Lionel Torres, **ReCoSoC08** , July 2008, Barcelona, SPAIN. [Paper] [Slides]

Others

Several directed/co-directed Master Thesis at Menta, Exhibition Booths, FPL10 Benchmarks, active participation in industrial forums discussions EEtimes/EEtimes-like etc.

Brief CV & Contact

Born: 2nd August 1980 in Rawalpindi (PAKISTAN)

Apr. 1998 – Sept. 2002: BS Electrical Engineering, U.E.T Taxila, PAKISTAN

Nov. 2002 – Sept. 2003: Embedded Systems Designer, AKSA-SDS, Islamabad, PAKISTAN

Oct. 2003 – Aug. 2006: MS Electrical Engineering (iCE-2003), TU-Darmstadt, GERMANY

Jun.2005 – Nov. 2005: Academic Research (Dynamic Reconf.), Fraunhofer ESK, Munich, GERMANY

Feb. 2006 – Jul. 2006: Academic Research (Homogeneous MPSoCs, NoCs), LIRMM, Montpellier, FRANCE

Sept. 2006 – Feb. 2011: at Menta SAS (embedded FPGAs), Montpellier, FRANCE

Dec. 2007 – Feb.2011: Industrial PhD (*this thing*), Menta + U. of Montpellier (LIRMM), FRANCE, defended thesis on 22 June 2011

Contact:

Email: sza_51@yahoo.com | syed-zahid.ahmed@cifre.org

Web: www.linkedin.com/in/SyedZahidAhmed

References

The references are split up into six categories to facilitate browsing of the bibliography in an easy and well differentiated way, benefits are/will be self evident. In addition to categorization, particular emphasis is put on the instant accessibility of the references in accordance with the modern electronic media styles/trends (things are/must/usually be click-drop away), enormously facilitating the readers/researches. The outline of main sections is as follows. 1 provides references to literature of FPGA vendors, 2 provides industrial articles/press releases, 3 provides patents of FPGA vendors, 4 provides references from academia, 5 gives general references (books etc.) and finally S outlines the self references (part of thesis contributions).

It might be the first customized domain-specific bibliography style of its kind in a PhD report.

1- FPGA Vendors

XILINX

1- WP369 - Extensible Processing Platform Ideal Solution for a Wide Range of Embedded Systems (2010): http://www.xilinx.com/support/documentation/white_papers/wp369_Extensible_Processing_Platform_Overview.pdf
2- Xilinx28nm Extensible Processing Platforms (Zynq-7000) devices with Hard ARM (2011) http://www.xilinx.com/technology/roadmap/processing-platform.htm
3- Virtex-6 FPGA Configurable Logic Block User Guide (2009): http://www.xilinx.com/support/documentation/user_guides/ug364.pdf
4- WP298 - Power Consumption at 40 and 45 nm (2009): http://www.xilinx.com/support/documentation/white_papers/wp298.pdf
5- WP246 - Power Consumption in 65 nm FPGAs (2007): http://www.xilinx.com/support/documentation/white_papers/wp246.pdf
6- WP223 – Power vs. Performance: The 90nm Inflection Point (2006): http://www.xilinx.com/support/documentation/white_papers/wp223.pdf
7- WP221 - Static Power and the Importance of Realistic Junction Temperature Analysis (2005): http://www.xilinx.com/support/documentation/white_papers/wp221.pdf
8- Virtex-II Complete Data Sheet (2007): http://www.xilinx.com/support/documentation/data_sheets/ds031.pdf
9- The Stacked Silicon Interconnect technology (2010): http://www.xilinx.com/technology/roadmap/ssi-technology.htm
10- WP380_Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency (2010): http://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf
11- WP312_Xilinx Next Generation 28 nm FPGA Technology Overview: http://www.xilinx.com/support/documentation/white_papers/wp312_Next_Gen_28_nm_Overview.pdf

ALTERA

12- Introducing Innovations at 28 nm to Move Beyond Moore's Law (2010) : http://www.altera.com/literature/wp/wp-01125-stxv-28nm-innovation.pdf
13- Stratix V Device Handbook (2010): http://www.altera.com/literature/hb/stratix-v/stratix5_handbook.pdf
14- Stratix IV Device Handbook (2010): http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf

15- 40-nm Stratix IV FPGAs and HardCopy IV ASICs (2008): http://www.altera.com/literature/br/br-stratix-iv-hardcopy-iv.pdf
16- 40-nm FPGA Power Management and Advantages (2008): http://www.altera.com/literature/wp/wp-01059-stratix-iv-40nm-power-management.pdf
17- 40-nm FPGAs: Architecture and Performance Comparison (2008): http://www.altera.com/literature/wp/wp-01088-40nm-architecture-performance-comparison.pdf
18- Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison (2007): http://www.altera.com/literature/wp/wp-01007.pdf
19- Stratix III Programmable Power (2007): http://www.altera.com/literature/wp/wp-01006.pdf
20- FPGA Architecture (2006): http://www.altera.com/literature/wp/wp-01003.pdf
21- Fracturable FPGA Logic Elements (Mike Hutton, David Lewis et al, TVLIS 2006): http://www.altera.com/literature/cp/cp-01006.pdf
22- Stratix II Logic and Routing Architecture (David Lewis, et al, FPGA-2005): http://www.altera.com/literature/cp/cp-01005.pdf
23- How Fast is the Fastest FPGA? Stratix® III Performance Capabilities (2007): http://www.techonline.com/learning/webinar/198500647?queryText=fpga
24- How to design an FPGA architecture tailored for efficiency and performance (by Seyi Verma EETimes article, 2007): http://www.eetimes.com/design/automotive-design/4015097/How-to-design-an-FPGA-architecture-tailored-for-efficiency-and-performance
25- Compare Stratix III and Virtex-5 Core Power Consumption (by Seyi Verma, webcast video): http://www.altera.com/education/webcasts/videos/videos-stratixiii-core-power.html
26- The Stratix Routing and Logic Architecture (David Lewis, Vaughn Betz et al, FPGA-2003)
27- Strategic Considerations for Emerging SoC FPGAs (2011): http://www.altera.com/literature/wp/wp-01157-embedded-soc.pdf

ACTEL (now Microsemi)

28- Actel Smart Fusion mixed signal FPGAs: http://www.actel.com/products/smartfusion
29- Smart Fusion Data Sheet (2010): http://www.actel.com/documents/SmartFusion_DS.pdf
30- IGLOO Low-Power Flash FPGAs Datasheet (2009): http://www.actel.com/documents/IGLOO_DS.pdf
31- Competitive Programmable Logic Power Comparison White Paper (2008): http://www.actel.com/documents/Power_Comparison_WP.pdf

Others

32- Lattice FPGA devices: http://www.latticesemi.com/products/fpga/index.cfm?source=sidebar
33- SiliconBlue: iCE65 L-Series Data Sheet (2010): http://www.siliconbluetech.com/media/downloads/iCE65Datasheet.pdf
34- SiliconBlue: DiePlus Advantage Data Sheet (2010): http://www.siliconbluetech.com/media/downloads/DiePlusAdvantageDatasheet.pdf
35- Raptor™ FPGA Product Information Brief (2009): http://www.aboundlogic.com/Submit/Submit.php?docid=Raptor_FPGAs_PB001.pdf
36- Achronix Speedster product brief: http://www.achronix.com/speedster-product-brief.html
37- Achronix Speedster 22i based on intel 22nm: http://www.achronix.com/products/speedster22i.html
38- WP001_Introduction to Achronix FPGAs white paper (2008): http://www.achronix.com/achronix-picopipe-white-paper.html
39- ABAX 3PLD product brochure (2010): http://www.tabula.com/products/Abax_ProductBrochure.pdf
40- Tabula Spacetime technology (2010): http://www.tabula.com/technology/technology.php

2- Industry (General)

1- Semiconductor Industry Market leaders, http://en.wikipedia.org/wiki/Semiconductor_sales_leaders_by_year
2- Gorden E. Moore, "Cramming more components onto integrated circuits", 1965, (Moore's law paper)
3- Vaughn Betz (Altera), "FPGA Challenges and Opportunities at 40nm and Beyond", FPL2009 keynote
4- Vaughn Betz (Altera), "FPGAs at 28nm: Meeting the Challenge of Modern Systems-on-a-Chip", FPL2010 keynote
5- Patrick Lysaght (Xilinx), "The Field Programmable Logic Perspective", FPL2010 keynote
6- Steve Trimberger (Xilinx), "Redefining the FPGA", FPL2007 keynote
7- Mark Dickinson (Altera), "System-Level Design for FPGAs", FPL2007 keynote
8- Ajay Bhatt (intel), "Accelerating with Many-cores & Special Purpose Hardware", FPL2007 keynote
9- PANEL: Will power kill FPGAs?, FPGA2006, Monterey California http://conferences.ece.ubc.ca/isfpga2007/www/2006/panel.html
10- PANEL:CMOS vs Nano, Comerades or Rivals?,FPGA2009 http://www.ece.wisc.edu/~kati/fpga2009/FPGA2009panel.html
11- Mike Butts of Ambric Inc.: ESC-SV 2008 Special: Multicore and Massively Parallel Platforms and Moore's Law Scalability. http://www.techonline.com/learning/techpaper/212202148
12- Mike Butts: Fundamentals of Multicore Processing (intel sponsored) http://www.techonline.com/learning/course/212002400
13- Kenton Williston: Massively parallel processors: Who's still alive? .January 2009. http://www.dspdesignline.com/blogs/212700925
14- Olivier Coudert: Sept. 2009: Why FPGA startups keep failing. www.ocoudert.com/blog/2009/09/15/why-fpga-startups-keep-failing
15- Walden Rhines (2010): Viewpoint: Is semiconductor industry consolidation inevitable? http://www.eetimes.com/electronics-news/4088277/Viewpoint-Is-semiconductor-industry-consolidation-inevitable-
16- Xilinx Puts ARM core into its FPGAs: ESC 2010 Silicon Valley www.eetimes.com/news/latest/showArticle.jhtml?articleID=224600583
17- (March 2010, Altera: Industry Faces Platform Collision) : http://www.eetimes.com/news/semi/showArticle.jhtml?articleID=223101261
18- Junko Yoshida, 2010: 5 reasons why Samsung scares Japan: http://www.eetimes.de/en/5-reasons-why-samsung-scares-japan.html?cmp_id=7&news_id=222902107
19- Junko Yoshida 2008: How Samsung out-hustled Japan Inc.: http://www.eetimes.com/electronics-news/4079289/How-Samsung-out-hustled-Japan-Inc--item-1 / http://www.signallake.com/innovation/SamsungOuthustledJapan102008.pdf
20- Dylan McGrath (2010): A dissenting opinion on the 'programmable imperative': http://www.eetimes.com/electronics-blogs/programmable-logic-designline-blog/4033356/A-dissenting-opinion-on-the-programmable-imperative-
21- TSMC embedded NVM roadmap: http://www.tsmc.com/download/english/a05_literature/08_TSMC_Embedded_Non-Volatile_Memory.pdf
22- Kilopass NVM technologies comparison (2009): http://www.kilopass.com/technology/embedded-nvm-comparison - /- http://www.kilopass.com/wp-content/uploads/2010/04/comparison_of_embedded_nvm.pdf
23- FlexEOS Embedded FPGAs from M2000 Now Production Ready in 0.13um (2004): http://www.design-reuse.com/news/7965/flexeos-embedded-fpgas-m2000-0-13um.html
24- M2000 Intros Largest 90nm eFPGA (2005): http://www.design-reuse.com/news/9614/m2000-intros-largest-90nm-efpga.html

25- M2000 Opens Silicon Valley Office Serving eFPGA Customers Throughout North America (2005): http://www.design-reuse.com/news/9732/m2000-opens-silicon-valley-office-serving-efpga-customers-throughout-north-america.html
26- M2000 unveils optimized eFPGA architecture for DSP functions (2005): http://www.design-reuse.com/news/9858/m2000-optimized-efpga-architecture-dsp-functions.html
27- STMicroelectronics Introduces Highly Integrated Microcontroller with *Embedded Programmable Logic* for Wireless Infrastructure Applications (2005): http://www.st.com/stonline/press/news/year2005/p1416h.htm
28- Multi-purpOse dynamically Reconfigurable Platform for intensive HEterogeneoUS processing (MORPHEUS). www.morpheus-ist.org
29- Catherine Gross (2006): Embedded FPGA to reach 65-nm in 2007, says M2000: http://www.eetimes.com/electronics-news/4185518/Embedded-FPGA-to-reach-65-nm-in-2007-says-M2000
30- Peter Clarke (2007): Newly-funded M2000 preps FPGA product family: http://www.eetimes.com/electronics-news/4189612/Newly-funded-M2000-preps-FPGA-product-family
31- Dylan McGrath (2009): Where are they now? M2000: http://www.eetimes.com/electronics-blogs/programmable-logic-designline-blog/4033108/Where-are-they-now-M2000
32- FPGA startup crunch: Abound banks on roots, density (2009): http://www.eetimes.com/electronics-news/4183338/FPGA-startup-crunch-Abound-banks-on-roots-density
33- Abound claims dense interconnect is key to Raptor FPGA (2009): http://eetimes.eu/en/abound-claims-dense-interconnect-is-key-to-raptor-fpga?cmp_id=7&news_id=217100249
34- Atmel CAP Microcontrollers (130nm): http://www.atmel.com/products/AT91CAP
35- IBM Licenses Embedded FPGA Cores from Xilinx for Use in SoC ASICs (2002): http://www.xilinx.com/support/documentation/white_papers/wp164.pdf
36- IBM Cu-08 ASIC : https://www-01.ibm.com/chips/techlib/techlib.nsf/products/ASIC_Cu-08
37- Embeddable, reprogrammable IP cores available from Actel on chartered 0.18-micron process (2001, Vericore eFPGAs of Actel): http://www.design-reuse.com/news/481/embeddable-reprogrammable-ip-cores-available-actel-chartered-0-18-micron-process.html
38- Actel moves to embed FPGA in standard and custom chips (2001): http://www.eetimes.com/electronics-news/4156884/Actel-moves-to-embed-FPGA-in-standard-and-custom-chips
39- Actel Introduces Varicore EPGA (Embedded FPGA) IP Cores (2001): http://www.wirelessdesignonline.com/article.mvc/Actel-Introduces-Varicore-EPGA-Embedded-FPGA-0001?VNETCOOKIE=NO
40- Actel 2006 Annual report (hints and details of abandoned businesses of SRAM FPGAs and Varicore eFPGAs): www.actel.com/documents/ir/reports/AnnualReport2006.pdf
41- Adaptive Silicon claims to cut cost of embedded programmable logic in SoC designs (2001): http://www.design-reuse.com/news/878/adaptive-silicon-claims-cut-cost-embedded-programmable-logic-soc-designs.html
42- Adaptive Silicon's MSA 2500 Programmable Logic Core TSMC Test Chips Are Fully Functional (2001): http://www.design-reuse.com/news/661/adaptive-silicon-msa-2500-programmable-logic-core-tsmc-test-chips-fully-functional.html
43- LSI Logic ASICs to add programmable-logic cores (1999): http://www.eetimes.com/electronics-news/4038331/LSI-Logic-ASICs-to-add-programmable-logic-cores
44- Leopard Logic and TSMC enable configurable silicon platforms based on hyperblox FP embedded FPGA cores (2002): http://www.design-reuse.com/news/4354/leopard-logic-tsmc-configurable-silicon-platforms-hyperblox-fp-embedded-fpga-cores.html
45- Dylan McGrath (2010, on TierLogic), FPGA startup: Process tech eases ASIC migration : http://www.eetimes.com/electronics-news/4088048/FPGA-startup-Process-tech-eases-ASIC-migration

46- Altera (M. Hutton and R. May), "Programmable Solutions for Automotive Systems", Invited Paper DATE06. http://sites.google.com/site/mhutton1/2006_DATE_MH_auto.pdf?attredirects=0
47- Actel (S. Kaptanoglu, G. Bakker, A. Kundu, I. Corneillet), "A new high density and very low cost reprogrammable FPGA architecture", conference paper FPGA-99
48- Intel Stellarton, Atom+Altera FPGA chip (IDF 2010): http://newsroom.intel.com/servlet/JiveServlet/download/1186-26-2523/Day2_IDF_Keynote_Davis.pdf
49- Rick Merritt (2010), Intel packs Atom into tablets, set-tops, FPGAs: http://www.eetimes.com/electronics-news/4207694/Intel-packs-Atom-into-tablets--set-tops--FPGAs
50- Clive Maxfield (2010), Altera launches major embedded initiative: http://www.eetimes.com/electronics-news/4209541/Altera-launches-major-embedded-initiative
51- Xilinx Acquires AutoESL to Enable Designer Productivity and Innovation With FPGAs and Extensible Processing Platform (2011): http://press.xilinx.com/phoenix.zhtml?c=212763&p=irol-newsArticle&ID=1521536&highlight=

3- Patents of FPGA Vendors

All of the referred patents can be downloaded (PDF) for free using European Patent Office search engine (http://ep.espacenet.com), each referred patent is hyperlinked to its respective bibliographic data page. The year mentioned in citations is the year when patent was published/granted (not filing year). Patents which are more cumulative in terms of references to previous works are preferred to be listed as they give better ease to researchers to navigate other and past works (is relatively harder for patents compared to publications). Furthermore for giant vendors, the authors with several patents are also mentioned (Area of research, total patents), to further help navigating research works ease based on author name.

The patents survey list covers majority (but not all) of FPGA vendors. FPGA design has several levels of complexity/issues (architecture, CAD, layout, configuration, security, specialized blocks, verification etc.) which are heavily addressed and patented by the vendors. This research work mainly focused on basic fundamentals of the architectural innovations (logic block, routing architecture etc.), distinct innovations of emerging startup companies and cutting edge futuristic work directions of FPGA leaders.

Furthermore this patent survey and the way it is presented contributes additionally for researchers by giving an example of start point of how-to access and analyze patents which are vital parts of state of the art research but often ignored by researchers due to their complex organization, writing theme (harder than publication), access, browsing infrastructure etc. compared to researcher-friendly publications/conferences.

Xilinx (All Patents List, 3000+)

Some (not limited!) notable authors/co-authors: Steven P. Young (Architecture; 150+), Stephen M. Trimberger (Configuration, Security, Architecture; 150+)

1- Ross Freeman, "Configurable electrical circuit having configurable logic elements and configurable interconnects", US4870302, 1989 (*Credited FPGA Invention Patent*)
2- Steven P. Young, Venu M. Kondapalli, Martin L. Voogel, "PLD lookup table including transistors of more than one oxide thickness", US7053654B1, 2006
3- Steven P. Young, Venu M. Kondapalli, Ramakrishna K. Tanikella, "Six-input look-up table for use in a field programmable gate array", US7061271B1, 2006
4- Steven P. Young, Venu M. Kondapalli, Ramakrishna K. Tanikella, "Six-input look-up table and associated memory control circuitry for use in a field programmable gate array", US7075332B1, 2006
5- Venu M. Kondapalli, Trevor J. Bauer, Manoj Chirania, Philip D. Costello, Steven P. Young, "Programmable lookup table with dual input and output terminals in shift register mode", US7215138B1 , 2007

6- Venu M. Kondapalli, Trevor J. Bauer, Manoj Chirania, Philip D. Costello, Steven P. Young, "Programmable lookup table with dual input and output terminals in RAM mode", US7265576B1, 2007
7- Bernard J. New, Richard A. Carberry, "Programmable logic device having configurable logic blocks with user-accesible input multiplexers", US6292019B1, 2001
8- Alireza S. Kaviani, "FPGA with improved structure for implementing large multiplexers", US6556042B1, 2003
9- Steven P. Young, Trevor J. Bauer, Manoj Chirania, Venu M. Kondapalli, "Programmable logic block with dedicated and selectable lookup table outputs coupled to general interconnect structure", US7375552B1, 2008
10- Danesh Tavana, Wilson K. Yee, Victor A. Holen, "FPGA architecture with repeatable titles including routing matrices and logic matrices", US5883525A , 1999 (the detailed global arch ideas)
11- Steven P. Young, Kamal Chaudhary, Trevor J. Bauer, "FPGA repeatable interconnect structure with hierarchical interconnect lines", US5914616A , 1999
12- Steven P. Young, Trevor J. Bauer, Kamal Chaudhari, Sridhar Krishnamurthy, "FPGA repeatable interconnect structure with bidirectional and unidirectional interconnect lines", US5942913A , 1999
13- Steven P. Young, "Expandable interconnect structure for FPGAs", US6396303B1, 2002
14- Steven P. Young, "Integrated circuit with programmable routing structure including diagonal interconnect lines", US7276934B1, 2007
15- Steven P. Young, "Integrated circuit with programmable routing structure including straight and diagonal interconnect lines", US7279929B1, 2007
16- Steven P. Young, "Columnar Floorplan", US7557610B2, 2009
17- Steven P. Young, "Efficient tile layout for a programmable logic device", US7274214B1, 2007
18- Trevor J. Bauer, Steven P. Young, "Formation of columnar application specific circuitry using a columnar programmable logic device", US7478359B1, 2009, (structured-asic-like scenario)
19- Steven P. Young, "FPGA architecture having RAM blocks with programmable word length and width and dedicated address and data lines", US5933023A, 1999
20- Raymond C. Pang, Steven P. Young, "Block RAM having multiple configurable write modes for use in a Field Programmable Gate Array", US6373779B1, 2002
21- Prasad Rau, Atul V. Ghia, Suresh M. Menon, "Configuration memory architecture for FPGA", US6501677B1, 2002
22- David P. Schultz, Lawrence C. Hung, F. Erich Goetting, "Method & structure for configuring FPGAs", US6204687B1, 2001
23- Stephen M. Trimberger, "Programmable logic device and method of configuration", US7199608B1, 2007
24- Stephen M. Trimberger, "Integrated circuit with circuitry for overriding a defective configuration memory cell", US7187597B1, 2007
25- Khue Dong, Stephen M. Trimberger, Alok Mehrotra, "Programmable single buffered six pass transistor configuration", US5600264A, 1997
26- Raymond C. Pang, Walter N. Sze, John M. Thendean, Stephen M. Trimberger, Jenifer Wong, "Programmable logic device with method of preventing readback", US6981153B1, 2005
27- Bernard J. New, Steven P. Young, "Method and apparatus for incorporating a multiplier into an FPGA", US2002057104A1 , 2002
28- Stephen M. Douglas, Steven P. Young, Nigel G. Herron, Mehul R. Vashi, Jane W. Sowards, "Programmable gate array having interconnecting logic to support embedded fixed logic circuitry", US6798239B2 , 2004 (PowerPC/like)
29- Stephen M. Trimberger, Austin H. Lasia, "FPGA with time-multiplexed interconnect", US7268581B1, 2007
30- Steven P. Young, "Integrated circuits with novel handshake logic", US7605604B1, 2009

31- Steven P. Young, "Integrated circuits with bus-based programmable interconnect structures", US7635989B1 , 2009
32- Steven P. Young, "Pipelined unidirectional programmable interconnect in an integrated circuit", US7759974B1, 2010
33- Arifur Rahman, Stephen M. Trimberger, Bernard J. New, "Integrated circuit with through-die via interface for die stacking", US7518398B1, 2009
34- Raghunandan Chaware, Arifur Rahman, "Low cost bumping and bonding method for stacked die", US7863092B1, 2011

Altera (All Patents List, 2800+)

Some (not limited!) notable authors/co-authors: David Lewis (Architecture; 70+), Vaughn Betz (CAD, Architecture; 50+), Michael Hutton (CAD; 40+)

35- Robert F. Hartmann, Sau-Ching Wong, Yiu-Fai Chan, Jung-Hsing Ou, "Programmable Logic Array device using EPROM technology", US4774421A, 1988 (*Among Altera foundation patents*)
36- David Lewis, Bruce Pedersen, Sinan Kaptanoglu, Andy Lee, "Fracturable lookup table and logic element", US7800401B1, 2010
37- Sinan Kaptanoglu, David Lewis, Bruce Pedersen, "Fracturable incomplete look up table area efficient logic elements", US7030650B1, 2006
38- David M. Lewis, Paul Leventis, Andy L. Lee, Henry Kim, Bruce Pedersen, Chris Wysocki, Christopher F. Lane, Alexander Marquardt, Vikram Santurkar, Vaughn Betz, "Versatile logic element and logic array block", US7671626B1, 2010
39- David Lewis, "Programmable logic device having logic modules with improved register capabilities", US7459932B1, 2008
40- Ketan Padalia, Kimberly Bozman, Vaughn Betz, "Techniques for grouping circuit elements into logic blocks", US7707532B1, 2010
41- Paul Leventis, David Lewis, "Flexible routing resources in a programmable logic device", US7098687B1, 2006
42- David Lewis, David Cashman, "Programmable logic device having logic array block interconnect lines that can interconnect logic elements in different logic blocks", US7456653B2, 2008
43- David M. Lewis, Paul Leventis, Andy L. Lee, Brian D. Johnson, Richard Cliff, Srinivas T. Reddy, Christopher F. Lane, Cameron R. McClintock, Vaughn Betz, Chris Wysocki, Alexander R. Marquardt, "Routing architecture for a programmable logic device", US6970014B1, 2005
44- Vaughn Betz, Jonathan Rose, "Heterogeneous interconnection architecture for programmable logic devices", US6828824B2, 2004
45- Michael D. Hutton, Bruce Pedersen, Sinan Kaptanoglu, David Lewis, Tim Vanderhoek, "Organization of logic modules in programmable logic devices", US7368944B1, 2008
46- Tim Vanderhoek, Vaughn Betz, David Cashman, David Lewis, Michael Hutton, "Programmable logic device architecture and methods for implementing logic in those architectures", US7716623B1, 2010
47- Michael D. Hutton, David Lewis, "Programmable routing structures providing shorter timing delays for input/output signals", US7312633B1, 2007
48- Richard G. Cliff, Srinvas T. Reddy, Andy L. Lee, David Lewis, "Multiple size memories in a programmable logic device", US7236008B1, 2007
49- David Lewis, Vaughn Betz, Irfan Rahim, Peter McElheny, Yow-Juang W. Liu, Bruce Pedersen, "Apparatus and methods for adjusting performance of integrated circuits", US2009289696A1, 2009
50- David Lewis, "Low power routing multiplexers", US2006256781A1, 2006
51- David Lewis, "Merged logic element routing multiplexer", US7215141B2, 2007
52- David Lewis, "High speed techniques for simulating circuits", US7283942B1, 2007

53- Christopher Lane, Ketan Zaveri, Hyun Yi, Giles Powel, Paul Leventis, David Jafferson, David Lewis, Triet Nguyen, Vikram Santurkar, Michael Chan, Andy Lee, Brian Johnson, David Cashman, "Programmable logic device with redundant circuitry", US6965249B2, 2005
54- David M. Lewis, Vaughn Betz, Paul Leventis, Michael Chan, Cameron R. McClintock, Andy L. Lee, Christopher F. Lane, Srinvas T. Reddy, Richard Cliff, "System and method for optimizing routing lines in a programmable logic device", US6895570B2, 2005

Actel (All Patents List, 800+)

Some (not limited!) notable authors/co-authors: Sinan Kaptanoglu (Architecture; 25+), Jonathan Greene (Architecure, Configuraition; 25+)

55- Abbas A. El Gamal, Khaled A. El-Ayat, Jonathan W. Greene, Ta-Pen R. Guo, Justin M. Reyneri, "Programmable Interconnect Architecture", US4873459A, 1989 (*Among Actel foundation patents*)
56- Wenyi Feng, Sinan Kaptanoglu, "FPGA architecture having two-level cluster input interconnect scheme without bandwidth limitation", US7545169B1, 2009
57- King W. Chan, William C. T. Shu, Sinan Kaptanoglu, Chi Fung Cheng, "Dedicated interface architecture for a hybrid integrated circuit", US7389487B1, 2008
58- Sheng Feng, Jung-Cheun Lien, Eddy C. Huang, Chung-Yuan Sun, Tong Liu, Naihui Liao, Weidong Xiong, "Tileable Field-Programmable gate array architecture", US2008238477A1, 2008
59- Jung-Cheun Lien, Sheng Feng, Tong Liu, "Field-Programmable Gate Array Architecture", US6774672B1, 2004
60- Sinan Kaptanoglu, "Architecture for routing resources in a Field Programmable Gate Array", US7579868B2, 2009
61- Sinan Kaptanoglu, "Block level routing architecture in a field programmable gate array", US7557611B2, 2009
62- Benjamin S. Ting, "Architecture and interconnect scheme for programmable logic circuits", US7646218B2, 2010
63- John E. McGowan, William C. Plants, Joel D. Landry, Sinan Kaptanoglu, Warren K. Miller, "Flexible, high-performance Static RAM architecture for Field Programmable Gate Array ", US5744980A, 1998
64- Greg Bakker, Khaled El-Ayat, Theodore Speers, Limin Zhu, Brian Schubert, Rabindranath Balasubramanian, Kurt Kolkind, Thomas Barraza, Venkatesh Narayanan, John McCollum, William C. Plants, "Programmable system on a chip", US7613943B2, 2009
65- Samuel W. Beal, Sinan Kaptonoglu, Jung-Cheun Lien, William Shu, King W. Chan, William C. Plants, "Enhanced Field Programmable Gate Array", US7755386B2, 2010 (FPGA+Fixed logic)
66- William C. Plants, "SRAM cell controlled by FLASH memory cell", US7408815B2, 2008
67- John McCollum, Hung-Sheng Chen, Frank Hawley, "Non-volatile programmable memory cell for programmable logic array", US7590000B2, 2009
68- Fethi Dhaoui, Zhigang Wang, John McCollum, Richard Chan, Vidyadhara Bellippady, "Radiation-tolerant flash-based FPGA memory cells", US7768317B1, 2010
69- Arunangshu Kundu, "Clock tree network in a field programmable gate array", US7545168B2, 2009
70- Jonathan W. Greene, John McCollum, Volker Hecht, "Circuits and methods for testing FPGA routing switches", US7804321B2, 2010
71- Jonathan W. Greene, Gregory Bakker, Vidyadhara Bellipady, Volker Hecht, Theodore Speers, "PLD providing soft wakeup logic", US2010156457A1, 2010

Abound Logic/M2000 (All Patents List, 15+)

72- Frederic Reblewski, Olivier Lepape, "A reconfigurable integrated circuit with a scalable architecture", US6594810B1, 2003

73- Olivier V. Lepape, " Reconfigurable integrated circuit with scalable architecture including one or more adders", US7498840B2, 2009
74- Olivier V. Lepape, "Reconfigurable integrated circuits with scalable architecture including a plurality of special function elements", US7768301B2, 2010
75- Frederic Reblewski, Cesar Douady, "Packet-oriented communication in reconfigurable circuit(s)", US7568064B2, 2009
76- Frederic Reblewski, Olivier Lepape, "Reconfigurable circuit with redundant reconfigurable clusters", US2010095147A1, 2010
77- Carl Ebeling, Frederic Reblewski, Olivier V. Lepape, Jean Barbier, "Crossbar device constructed with MEMS switches", US2010108479A1, 2010

Achronix (All Patents List, 8+)

78- Rajit Manohar, "Converting a synchronous circuit design into an asynchronous design", US2010005431A1, 2010
79- Rajit Manohar, "Automated conversion of synchronous to asynchronous circuit design representations", US2009319962A1, 2009
80- Rajit Manohar, Gregor, Martin, John Lofton Holt, "Synchronous to asynchronous logic conversion", US7739682B2, 2010
81- Rajit Manohar, Clinton W. Kelly, "Reconfigurable logic fabric for integrated circuits and systems and methods for configuraing reconfigurable logic fabrics", US2010013517A1, 2010
82- Rajit Manohar, Clinton W. Kelly, "Fault tolerant asynchronous circuits", US2010207658A1, 2010

Tabula (All Patents List, 70+)

83- Jason Redgrave, Teju Khubchandani, Brad Hutchings, Steven Teig, Hermann Schmitt, "Configuralbe IC with packet switching configuration network", WO2009131569A1, 2009
84- Herman Schmit, Michael Butts, Brad L. Hutchings, Steven Teig, "Method of mapping a user design defined for a user design cycle to an IC with multiple sub-cycle reconfigurable circuits", US7872496B2, 2011
85- Brad Hutchings, Herman Schmit, Jason Redgrave, "Hybrid Logic/Interconnect Circuit in a configurable IC", US2008100336A1, 2008
86- Steven Teig, Herman Schmit, Jason Redgrave, "Configurable IC having a routing fabric with storage elements", US7525344B2, 2009
87- Andre Rohe, Steven Teig, "Configurable Integrated circuit with built-in turns", US7737722B2, 2010

Tier Logic/Viciciv (All Patents List, 15+)

88- Raminda Udaya Madurawe, "Three dimensional integrated circuits", US2009039918A1, 2009
89- Raminda U. Madurawe, "Semiconductor devices fabricated with different processing options", US7759705B2, 2010
90- Raminda U. Madurawe, Peter Ramyalal Suaris, Henry White, "MPGA products based on a prototype FPGA", US7673273B2, 2010
91- Raminda U. Madurawe, "Programmable logic devices comprising time multiplexed programmable interconnect", US7759969B2, 2010
92- Raminda U. Madurawe, Nij Dorairaj, "Programmable logic based latches and shift registers", US7573294B2, 2009
93- Raminda U. Madurawe, "Look-up table structure with embedded carry logic", US7466163B2, 2008
94- Nij Dorairaj, "Using programmable latch to implement logic", US7602213B2, 2009

4- Academics

Publications are grouped under the name of research head in case of five or more publications from same author/co-author (names of Professors are in non-priority random order!). All publications/presentations (where applicable) are hyperlinked to the direct access provided by Professors or institutes web pages, to maintain equivalent click-to-get like form of this reference section like for industrial references and patents. Reference 4.1 (VPR book) is an exceptional alias of 5.1.

Prof. Jonathan Rose (Toronto Univ.), FPGA architecture and CAD: http://www.eecg.toronto.edu/~jayar

1- V. Betz, J. Rose, and A. Marquardt, Architecture and CAD for Deep-Submicron FPGAs, Kluwer Academic Publishers, February 1999. ISBN 0-7923-8460-1 {same as ref. 5.1of books section}
2- I. Kuon, R. Tessier and J. Rose "FPGA Architecture: Survey and Challenges", Foundations and Trends in Electronic Design Automation: Vol. 2: No 2, 2008, pp. 135-253
3- I. Kuon and J. Rose, "Measuring the Gap between FPGAs and ASICs" in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, NO. 2, Feb. 2007
4- J. Rose, "The Evolution of Architecture Exploration of Programmable Devices", FPL-2009 Keynote
5- J. Luu, I. Kuon, P.Jamieson, T. Campbell, A. Ye, M. Fang, and J. Rose "VPR 5.0: FPGA CAD and Architecture Exploration Tools with Single-Driver Routing, Heterogeneity and Process Scaling," in FPGA-2009
6- I. Kuon and J. Rose, "Automated Transistor Sizing for FPGA Architecture Exploration" in DAC '08, ACM/IEEE Design Automation Conference, June 2008, pp. 792-795
7- I. Kuon and J. Rose "Area and Delay Trade-offs in the Circuit and Architecture Design of FPGAs," in ACM Symposium on FPGAs, February 2008, pp. 149-158
8- E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density" , IEEE Trans. on VLSI, March 2004, pp. 288-298.
9- Andy Ye, J. Rose, "Using Bus-Based Connections to improve Field-Programmable Gate-Array Density for implementing Datapath Circuits", IEEE Trans on VLSI, May 2006, pp. 462-473
10- A. Marquardt, V. Betz and J. Rose, ``Speed and Area Tradeoffs in Cluster-Based FPGA Architectures", IEEE Transactions on VLSI Systems, February 2000, pp. 84 - 93.
11- V. Betz and J. Rose, ``Automatic Generation of FPGA Routing Architectures from High-Level Descriptions" , FPGA-2000
12- V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research" in 7th International Workshop on Field-Programmable Logic, London, August 1997, pp. 213-222.
13- The GILES/ATL Project Documents - Automated Layout of FPGAs from Architectural Specification

Prof. Guy Lemieux (UBC) , FPGA architecture and CAD: http://www.ece.ubc.ca/~lemieux

14- V. Aken'ova, G. Lemieux, R. Saleh, ``Soft++: An Improved Embedded FPGA Methodology for SoC Designs", *IEEE Transactions on VLSI*, 2007
15- J. Lamoureux, G. Lemieux, S. Wilton, ``GlitchLess: Dynamic Power Minimization in FPGAs through Edge Alignment and Glitch Filtering", *IEEE Transactions on VLSI*, November, 2008
16- E. Lee, G. Lemieux, S. Mirabbasi, ``Interconnect Driver Design for Long Wires in Field-Programmable Gate Arrays", *Journal of Signal Processing Systems*, Springer, 51(1), April 2008
17- P. Teehan, G. Lemieux, M. Greenstreet, "Towards Reliable 5Gbps Wave-pipelined and 3Gbps Surfing Interconnect in 65nm FPGAs", FPGA-2009. presentation
18- M. Tom, D. Leong, G. Lemieux, "Un/DoPack: Re-Clustering of Large System-on-Chip Designs with Interconnect Variation for Low-Cost FPGAs", *IEEE International Conference Computer-Aided Design*, San Jose, November 2006. presentation
19- G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and Single-Driver Wires in FPGA Interconnect", *IEEE International Conference on Field-Programmable Technology*, Brisbane, Australia, pp. 41-48, December 2004. **Best Paper Award.** presentation

20- D. Yeager, D. Chiu, G. Lemieux, ``Congestion Estimation and Localization in FPGAs: A Visual Tool for Interconnect Prediction", *International Workshop on System-Level Interconnect Prediction*, Austin, TX, March 2007. presentation
21- G. Lemieux, D. Lewis, ``Analytical Framework for Switch Block Design", *Field-Programmable Logic and Applications*, La Grande Motte, France, pp. 122-131, September 2002. presentation
22- G. Lemieux, D. Lewis, ``Checkerboard Switch Block Topologies for Routing Diversity", poster at *ACM/SIGDA International Symposium on FPGA*-2002. presentation (HTML)
23- G. Lemieux, D. Lewis, ``Circuit Design of FPGA Routing Switches", *ACM/SIGDA International Symposium on FPGAs*, Monterey, CA, pp. 19-28, February 2002. presentation (HTML only)
24- G. Lemieux, D. Lewis, ``Using Sparse Crossbars within LUT Clusters", *ACM/SIGDA International Symposium on FPGAs*, Monterey, CA, pp. 59-68, February 2001.

Prof. Steve Wilton (UBC), FPGA architecture and CAD: http://www.ece.ubc.ca/~stevew

25- Steven J.E. Wilton, ``Architecture and Algorithms for Field-Programmable Gate Arrays with Embedded Memory", PhD thesis, University of Toronto, 1997.
26- J. Lamoureux, S.J.E. Wilton, "FPGA Clock Network Architecture: Flexibility vs. Area and Power", FPGA-2006.
27- B. Quinton, S.J.E. Wilton, "Embedded Programmable Logic Core Enhancements for System Bus Interfaces", FPL-2007

28- B.R. Quinton, S.J.E. Wilton, "Programmable Logic Core Based Post-Silicon Debug For SoCs", 4th IEEE Silicon Debug and Diagnosis Workshop, Germany, May 2007. [slides]
29- N. Kafafi, K. Bozman, S.J.E. Wilton, ``Architectures and Algorithms for Synthesizable Embedded Programmable Logic Cores", in the ACM International Symposium on Field-Programmable Gate Arrays, Monterey, CA, Feb 2003, pp. 1-9. [slides]
30- K.K.W. Poon, S.J.E. Wilton, A. Yan, "A Detailed Power Model for Field-Programmable Gate Arrays", in ACM Transactions on Design Automation of Electronic Systems (TODAES), 2005
31- S.J.E. Wilton, N. Kafafi, J. Wu, K. Bozman, V. Aken'Ova, R. Saleh, "Design Considerations for Soft Embedded Programmable Logic Cores", IEEE Journal of Solid-State Circuits, 2005
32- T. Wong, S.J.E. Wilton, "Placement and Routing for Non-Rectangular Embedded Programmable Logic Cores in SoC Design", FPT-2004
33- J.C.H. Wu, V. Aken'Ova, S.J.E. Wilton, R. Saleh, ``SoC Implementation Issues for Synthesizable Embedded Programmable Logic Cores", in IEEE Custom Integrated Circuits Conference, 2003
34- S.J.E. Wilton, R. Saleh, ``Programmable Logic IP Cores in SoC Design: Opportunities and Challenges", in the IEEE Custom Integrated Circuits Conference, San Diego, CA, May 2001
35- M. Imran Masud, ``FPGA Routing Structures: A Novel Switch Block and Depopulated Interconnect Matrix Architecture", M.A.Sc. Thesis, December 1999.

Joint with Prof. Wayne Luk (Imperial College London): http://www.doc.ic.ac.uk/~wl

36- P. Jamieson, W. Luk, S.J.E. Wilton, G. Constantinides, "An Energy and Power Consumption Analysis of FPGA Routing Architectures", FPT-2009 (poster presentation)
37- C.T. Chow, L.S.M. Tsui, P.H.W. Leong, W. Luk, S. Wilton, "Dynamic Voltage Scaling for Commercial FPGAs", FPT-2005. **Best Paper Award**
38- S.J.E. Wilton, S-S. Ang, W. Luk, "The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays", FPL-2004. Included in Lecture Notes in Computer Science 3203, Springer-Verlag, pp. 719-728. **Best Paper Award** [slides]
39- S.J.E. Wilton, C.H. Ho, P.H.W. Leong, W. Luk, B. Quinton, "A Synthesizable Datapath-Oriented Embedded FPGA Fabric", FPGA-2007. [slides]

40- S.J.E. Wilton, B. Quinton, C.H. Ho, P.H.W. Leong, W. Luk, "A Synthesizable Datapath-Oriented Embedded FPGA Fabric for Silicon Debug Applications", in ACM Transactions on Reconfigurable Technology and Systems, vol. 1, no. 1, March 2008, pp 7.1-7.25

Prof. Jason Cong (UCLA), VLSI CAD, FPGA CAD, ESL: http://cadlab.cs.ucla.edu/~cong

41- D. Chen, J. Cong and P. Pan, "FPGA Design Automation: A Survey," Foundations and Trends in Electronic Design Automation, vol. 1, no. 3, pp. 195-330, Nov 2006
42- "Architecture and Synthesis for Power-Efficient FPGA's," (Invited Dinner Speech), 2004 IEEE Electronic Design Process Symposium (EDPS), Monterey, California, April, 2004
43- "Large-Scale Circuit Placement: Gap and Progress," University of Toronto Distinguished Lecture Series, University of Toronto, Ontario, Canada, March 24, 2005
44- "Is the Second Wave of HLS the One Industry Will Surf on?", DATE-2009 Panel
45- "Customizable Domain-Specific Computing", FPL-2009 (Keynote Speech)
46- J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", IEEE Trans. on Computer-Aided Design, January 1994. (1995 Circuit and System Society **Best Paper Award** in IEEE Transactions on CAD)
47- G. Chen and J. Cong, "Simultaneous Placement with Clustering and Duplication," ACM Transaction on Design Automation of Electronic Systems, vol. 11, no. 3, pp. 740-772, July 2006
48- F. Li, Y. Lin, L. He, D. Chen, and J. Cong, "Power Modeling and Characteristics of Field Programmable Gate Arrays," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, Issue 11, pp. 1712-1724, November 2005
49- F. Li, D. Chen, L. He, J. Cong, "Architecture Evaluation for Power-Efficient FPGAs," , FPGA-2003
50- F. Li, Y. Lin, L. He, and J.Cong, "Low-power FPGA using Pre-Defined Dual-Vdd/Dual-Vt Fabrics," ,FPGA-2004
51- D. Chen, J. Cong, F. Li, and L. He, "Low-Power Technology Mapping for FPGA Architectures with Dual Supply Voltages," FPGA-2004
52- C.T. Hsieh, J. Cong, Z. Zhang and S.C. Chang, "Behavioral Synthesis with Activating Unused Flip-Flops for Reducing Glitch Power in FPGA", ASP-DAC 2008

Prof. Habib Mehrez (LIP6), FPGA architecture and CAD: http://www-asim.lip6.fr/~habib

53- Husain Parvez, “Design and Exploration of Application-Specific Mesh-Based Heterogeneous FPGA Architectures”, Ph.D. Thesis, LIP6, 2010 , {Published as a book also}
54- Husain Parvez, Zied Marrakchi, Habib Mehrez, “Heterogeneous-ASIF: an application specific inflexible FPGA using heterogeneous logic blocks”, FPGA 2010
55- Umer Farooq, Husain Parvez, Zied Marrakchi, Habib Mehrez, “Exploration of Heterogeneous FPGA architectures”, ReCoSoC-10, 2010
56- Zied Marrakchi, Hayder Mrabet, Umer Farooq, and Habib Mehrez, “FPGA Interconnect Topologies Exploration”, International Journal of Reconfigurable Computing (IJRC), 2009
57- Hyder Mrabet, “Design and optimization of reconfigurable architectures: The FPGA Family”, Ph.D. Thesis, LIP6, 2009
58- Zied Marrakchi, “Exploration and Optimization of Tree-based FPGA Architectures”, Ph.D. Thesis, LIP6, 2008

Prof. Rajit Manohar (Cornell University/Achronix), Asynchronous FPGAs: http://vlsi.cornell.edu/~rajit

59- Song Peng, David Fang, John Teifel, Rajit Manohar, “Automated synthesis for asynchronous FPGAs”, FPGA-2005
60- John Teifel, Rajit Manohar, “Highly pipelined asynchronous FPGAs”, FPGA-2004
61- John Teife, Rajit Manohar, “Programmable Asynchronous Pipeline Arrays”, FPL-2003

62- David Fang, John Teifel, Rajit Manohar, "A High-Performance Asynchronous FPGA: Test Results", FCCM-2005
63- Patents list (http://vlsi.cornell.edu/~rajit/pub_all.html)

Prof. Reiner Hartenstein (Univ. Kaiserslautern), Reconfigurable Computing: http://hartenstein.de

64- R. Hartenstein, "A Decade of Reconfigurable Computing: a Visionary Retrospective", DATE-2001 embedded tutorial, [Paper]
65- R. Hartenstein, "Reconfigurable Computing: the Roadmap to a New Business Model - and its Impact on SoC Design", SBCCI-2001 keynote, [Paper]
66- R. Hartenstein, "Enabling technologies for reconfigurable computing", Part 4, Tampere, 2001 lecture, includes Prof. Jonathan Rose (when will FPGAs kill ASICs slides also); Other: [Part1][Part2][Part3]
67- R. Hartenstein, "Reconfigurable HPC: Torpedoed by Deficits in Education?",RHPC-2004 keynote
68- R. Hartenstein, "From organic computing to reconfigurable computing", PASA-2006 keynote
69- R. Hartenstein, "The von Neumann Syndrome", 2007 invited paper in SVMS Delft, 2007
70- R. Hartenstein, "Multicore programming and the CS education dilemma", MPSoC-2009 keynote
71- R. Hartenstein, "A visionary retrospective about reconfigurable computing education ", ReCoSoC-10
72- R. Hartenstein, "Reconfigurable computing: boosting Software education for the Multicore era", SPL-2010 keynote

LIRMM (Univ. of Montpellier) MRAM work**; Prof. Lionel Torres:** http://www.lirmm.fr/~torres

73- N. Bruchon, "Evaluation, validation and design of hybrid CMOS Non Volatile emerging technology cells for dynamically reconfigurable fine grain architecture", PhD thesis, Univ. of Montpellier, 2007
74- Y. Guillemenet, L. Torres, G. Sassatelli, N. Bruchon, "On the use of Magnetic RAMs in Field Programmable Gate Arrays", IJRC volume 2008, Article ID 723950, 9 pages
75- Y. Guillemenet, L. Torres, G. Sassatelli, I. Hassoune, "A non-volatile run-time FPGA using thermally assisted switching MRAMs", FPL-2008
76- Y. Guillemenet, L. Torres, G. Sassatelli, "Non-volatile run-time field-programmable gate arrays structures using thermally assisted switching magnetic random access memories", IET 2010
77- L. V. Cargnini, Y. Guillemenet, L. Torres, G. Sassatelli, "Improving the Reliability of a FPGA using Fault-Tolerance Mechanism Based on Magnetic Memory (MRAM)", ReConFig-10
78- Researchers present MRAM based FPGA (2009 EEtimes Intervew by Anne-Francoise Pele): http://www.eetimes.com/electronics-news/4084935/Updated-Researchers-present-MRAM-based-FPGA-architecture
79- First MRAM based FPGA taped-out (2010): http://www.eetimes.com/electronics-products/other/4200035/First-MRAM-based-FPGA-taped-out
80- The SPIN project website: http://www.lirmm.fr/SPIN

MISC. (FPGAs/Reconfigurable Hardware)

81- R. Njuguna, R. Jain, "A Survey of FPGA Benchmarks", Washington University in St. Louis, 2008
82- Saeyang Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0", Microelectronic Center of North Carolina (MCNC), 1991
83- Prof. Alan Mishchenko (SIS/ABC tools research): http://www.eecs.berkeley.edu/~alanmi
84- Prof. Jason Anderson (FPGA power challenges research): http://www.eecg.toronto.edu/~janders
85- Jason H. Anderson, "Power optimization and prediction techniques for FPGAs," Ph.D. Thesis, Department of Electrical and Computer Engineering, University of Toronto, 2005.
86- R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool", Proc. CAV'10, Springer, LNCS 6174, pp. 24-40. PDF
87- B. Neumann, T. von Sydow, H. Blume, T. G. Noll, "Design Flow for Embedded FPGAs Based on a Flexible Architecture template", DATE2008

88- Andre DeHon, "Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you don't really want 100% LUT utilization)", FPGA-99
89- Claudio Brunelli, Fabio Garzia, Jari Nurmi, Fabio Campi, Damien Picard: "Reconfigurable hardware: The holy grail of matching performance with programming productivity". FPL-2008
90- Juergen Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators, Ph. D. Dissertation 1997 - foto - URL
91- J. Becker, A. Thomas, M. Vorbach, and V. Baumgarte, "An Industrial/Academic Configurable System-On-Chip Project (CSoC): Coarse-Grain XPP-/Leon-Based Architecture Integration", DATE2003
92- H. Fan, J. Liu, Y.L. Wu, and C.C. Cheung, "On Optimal Hyper-universal and Rearrangeable Switch Box Designs", in IEEE Transactions on CAD, 2003
93- L. Zhou, C.C. Cheung and Y.L. Wu, "What if Merging Connection and Switch Boxes --- an Experimental Revisit on FPGA Architectures", ICCCAS, 2004. (**Best Paper Award**)
94- H. Fan, Y.L. Wu, C.C. Cheung and J. Liu, "On Optimal Irregular Switch Box Designs", FPL-2004. Slides
95- Estrin, G., "Organization of Computer Systems—The Fixed Plus Variable Structure Computer," Proc. Western Joint Computer Conf., Western Joint Computer Conference, New York, 1960, pp. 33–40
96- Versatile Place and Route for Hybrid FPGAs project: http://cc.doc.ic.ac.uk/projects/VPH
97- Totem Project (Univ. of Washington): http://ee.washington.edu/faculty/hauck/Totem
98- S. Hauck, K. Compton, K. Eguro, M. Holland, S. Phillips, A. Sharma, "Totem: Domain-Specific Reconfigurable Logic", submitted to *IEEE Transactions on VLSI Systems*
99- K. Compton, Architecture Generation of Customized Reconfigurable Hardware , PhD Thesis, *Northwestern University, Department of ECE*, December 2003
100- K. Compton, S. Hauck, "Reconfigurable Computing: A Survery of Systems and Software", ACM Computing surveys, 2002
101- Larry McMurchie, Carl Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs", FPGA-1995
102- Sumanta chaudhuri, Asynchronous FPGA Architectures for Cryptographic Applications ,(Slides), Phd Thesis, May 15th 2009, Paris, FRANCE
103- Sumanta chaudhuri, "Diagonal tracks in FPGAs: a performance evaluation" ,(Slides). FPGA-2009
104- A. Lodi, M. Toma, F. Campi, A. Cappelli, R. Canegallo, R. Guerrieri, "A VLIW processor with reconfigurable instruction set for embedded applications", IEEE journal of Solid-State Circuits, 2003
105- J. R. Hauser, J. Wawrzynek, "Garp: a MIPS processor with a reconfigurable coprocessor", FCCM-97
106- S. C. Goldstein et al, "PipeRench: A reconfigurable architecture and compiler", IEEE Computer, 2000
107- Chen Chen, Roozbeh Parsa, Nishant Patil, Soogine Chong, Kerem Akarvardar, J Provine, David Lewis, Jeff Watt, Roger T. Howe, H.-S. Philip Wong, Subhasish Mitra, "Efficient FPGAs using nanoelectromechanical relays", FPGA-2010
108- Rikky Muller, Chintan Thakkar. "Use of Nano-Mechanical Relays for FPGA Power Reduction". 2008, http://inst.eecs.berkeley.edu/~cthakkar/project_home.html

5- Books & Others

Books

1- Vaughn Betz, Jonathan Rose, Alexander Marquardt, "Architecture and CAD for Deep-Submicron FPGAs", Springer, 1999
2- Ian Kuon, Jonathan Rose, "Quantifying and Exploring the Gap Between FPGAs and ASICs", Springer, 2009
3- Ian Kuon, Russell Tessier, Jonathan Rose , "FPGA Architecture: Survey and Challenges", Now Publishers Inc., 2008
4- Erik Brunvand, "Digital VLSI Chip Design with Cadence and Synopsys CAD Tools", Addison-Wesley, 2009
5- David Chinnery, Kurt Keutzer, "Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design", Springer, 2002
6- David Chinnery, Kurt Keutzer, "Closing the Power Gap between ASIC & Custom: Tools and Techniques for Low Power Design", Springer, 2010
7- Akio Morita, "Made in Japan: Akio Morita and Sony", NY E.P. Dutton & Co, 1986
8- Sea-Jin Chang, "Sony vs Samsung: The Inside Story of the Electronics Giants' Battle For Global Supremacy", Wiley, 2008
9- Merrill R. Chapman, "In Search of Stupidity, over 20 years of high-tech marketing disasters" , Apress, 2006
10- Jorge Cham, "Piled Higher & Deeper" (1997- Present): http://www.phdcomics.com

Companies/Organizations etc.

11- CIFRE Phd : http://www.anrt.asso.fr/fr/espace_cifre/accueil.jsp / English Brochure : http://www.anrt.asso.fr/fr/pdf/plaquette_cifre_complete_avril2009_GB.pdf
12- Menta SAS (embedded FPGAs): www.menta.fr , www.efpga.com
13- CMP (Circuits Multi-Projects) http://cmp.imag.fr
14- Qt (Nokia) Cross-platform application and UI framework: http://qt.nokia.com
15- Freescale's MRAM spun out: www.everspin.com/technology.html
16- Crocus MRAM technology: www.crocus-technology.com
17- MRAMs information/news: www.memorystrategies.com/report/focused/mram.htm
18- MRAMs information/news: www.mram-info.com
19- MRAM post processing above CMOS: http://www-leti.cea.fr
20- Universal Memories: http://en.wikipedia.org/wiki/Universal_memory
21- Magenetoresistive RAM (MRAM): http://en.wikipedia.org/wiki/MRAM
22- LEON3, Garisler research (Aeroflex): www.gaisler.com
23- SPARC V8 manual: www.sparc.com/standards/V8.pdf
24- OpenCores: http://opencores.org
25- SimpleScalar tools: www.simplescalar.com

General Knowledge

26- How to Have a Bad Career in Research/Academia (Prof. David A. Patterson, 2001): http://www.cs.berkeley.edu/~pattrsn/talks/BadCareer.pdf
27- Prof. Joe Wolfe, "How to write a PhD thesis": www.phys.unsw.edu.au/~jw/thesis.html
28- Dr. Steve Easterbrook, "How theses get written: some cool tips": www.cs.toronto.edu/~sme/presentations/thesiswriting.pdf
29- Prof. S. J. Levine, "Writing and Presenting your Thesis or Dissertation": www.learnerassociates.net/dissthes/dissguid.pdf

30- Dr. Keith Morgan, “Writing a PhD Thesis”: www.shintonconsulting.com/downloads/0703/NCL_Thesis_writing_07.ppt
31- Michel Thomas Foreign Language courses: www.michelthomas.co.uk
32- Bill Gates: Harvard University commencement speech, June 2007.[YouTube]
33- Steve Jobs: Stanford University commencement speech, June 2005. [YouTube]
34- J. K. Rowling: Harvard University commencement speech , June 2008. [YouTube]
35- Internet movie database: www.imdb.com
36- www.wikipedia.org
37- www.google.com

S- Self References

1- SoC Configurability- Balancing Manufacturing and R/D Costs. IP08 PANEL, Dec. 2008, Grenoble, FRANCE [Slides]
2- IP/SoC Prototyping. IP07 PANEL, Dec. 2007, Grenoble, FRANCE [Slides]
3- Syed Zahid Ahmed, Gilles Sassatelli, Lionel Torres, Laurent Rouge, "Survey of new trends in Industry for Programmable hardware: FPGAs, MPPAs, MPSoCs, Structured Asics, eFPGAs and new wave of innovation in FPGAs". FPL-10, Sept. 2010, Milan, ITALY. 3a [Paper] : 3b [SLIDES*]
4- Lionel Torres, Yoann Guillemenet, Syed Zahid Ahmed, "A Dynamic Reconfigurable MRAM based FPGA". ERSA-10 invited keynote paper, July 2010, Las Vegas, USA.
5- Yoann Guillemenet, Syed Zahid Ahmed, Lionel Torres, Alexandre Martheley, Julien Eydoux, Jean-Baptiste Cuelle, Laurent Rouge, Gilles Sassatelli, "MRAM based eFPGAs: programming and silicon flows, exploration environments, MRAM current state in Industry and its unique potentials for FPGAs". ReConFig09, Dec. 2009, Cancun, MEXICO. [Slides]
6- Syed Zahid Ahmed, Alexandre Martheley, Laurent Rouge, Julien Eydoux, Jean-Baptiste Cuelle, "FPGA Designer GUI Tools Suite: A complete hardware and software infrastructure for creating customizable eFPGA IP blocks of Menta". IPESC09, Dec. 2009, Grenoble, FRANCE. [D&R][Slides]
7- Syed Zahid Ahmed, Julien Eydoux, Laurent Rouge, Jean-Baptiste Cuelle, Gilles Sassatelli, Lionel Torres, "Exploration of power reduction and performance enhancement in LEON3 processor with ESL reprogrammable eFPGA in processor pipeline and as a co-processor". DATE-2009, April 2009, Nice, FRANCE. [Slides]
8- Syed Zahid Ahmed, Julien Eydoux, Michael Fernandez, Laurent Rouge, Gilles Sassatelli, Lionel Torres, "Power consumption reduction explorations in processors by enhancing performance using small ESL reprogrammable eFPGAs". ReConFig08, December 2008, Cancun, MEXICO. [Slides]
9- Syed Zahid Ahmed, Michael Fernandez, Gilles Sassatelli, Lionel Torres, "eFPGA architecture explorations: CAD and Silicon analysis of beyond 90nm technologies to investigate new dimensions of future innovations”, ReCoSoC08 , July 2008, Barcelona, SPAIN. [Slides]

****