**Automated Conversion from LUT-based FPGAs
to LUT-based MPGAs**

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

(DR.-ING.)

der Fakultät für Ingenieurwissenschaften
und Informatik der Universität Ulm

von

FRANCISCO JAVIER VEREDAS RAMIREZ

AUS MANRESA (SPANIEN)

1. Gutachter: Prof. Dr.-Ing. Hans-Jörg Pfeleiderer

2. Gutachter: Prof. Dr.-Ing. Ulf Schlichtmann

Amtierender Dekan: Prof. Dr. rer. nat. Helmuth Partsch

Ulm, 15.März 2007

to my mother, sister and nieces

Abstract

Field-programmable gate-arrays (FPGAs) are used for application-specific standard product (ASIC) prototyping or small volume products. In medium to large volume products, the prototyping design or the small volume product is converted to another integrated circuit (IC) structure such as mask-programmable gate arrays (MPGAs). MPGAs are of growing importance because of the increase of design cost, and turnaround times in ultra-deep submicron technologies which mostly impact ASICs. Several design methodologies have been proposed in recent years for converting an evaluated FPGA prototype-design into an MPGA. The MPGA design uses potentially less area, delay, and dynamic power consumption than the FPGA design.

In a conversion from an FPGA design, the engineer looks for a simple flow to minimize time-to-market. It is well known that the most time consuming process in an IC design is verification. Formal verification checks the functionality of the designed circuit. Physical verification checks the timing (and in some cases the power consumption) of the designed circuit. Formal verification is simplified if the same gate-level netlist of the FPGA is used in the conversion. Simplification of physical verification is possible by exploiting the regularity in the IC layout.

In this thesis, two new Look-Up Table-based (LUT-based) MPGAs are presented. The MPGA architectures preserve the logic of an FPGA, and have a regular routing interconnect scheme. The use of a regular routing architecture reduces the time for physical verification. The first MPGA presented,

r-MPGA, is a direct migration from an FPGA, i.e. the programmable interconnect points (PIPs) and the latches used for logic configuration are substituted by metalization. The second MPGA, Zelix MPGA, has a new routing architecture which only uses two metal layers for mask-programming. The conversion flow for the Zelix MPGA is presented. The conversion flow preserves the gate-level netlist, and re-use the placement. The re-use of the same gate-level netlist minimizes the time for formal verification. The resulting flow has a special routing tool, and buffer insertion algorithm for timing integrity. In addition, a simple methodology to estimate power consumption is presented. Power consumption is estimated using gate-level information. Comparison with transistor-level simulations shows a difference of 4.83% in the gate-level power consumption estimation.

In comparison with an FPGA, the area is reduced by 82% in the logic, and by up-to 64% in the whole device. To investigate the delay, and the power consumption, a set of application circuits has been mapped. Experimental investigations show that the critical path is reduced in average by 48% compared with the FPGA. The reduction of dynamic power consumption is 72% on average. Static power consumption on the Zelix MPGA is less than 1% of the total power consumption. The conversion flow is rather simple, and can map a designed circuit within one day.

Acknowledgements

First and foremost, I would like to thank my supervisors, Prof. Dr.-Ing. Hans-Jörg Pfeiderer (University of Ulm) and Michael Scheppler (Qimonda Flash GmbH). My honest and deeply respectful appreciations are for Prof. Dr.-Ing. Hans-Jörg Pfeiderer for all his detailed and insightful technical advice over more than three years. Thanks for the kind reception at his chair specially Oliver Pfänder and Mrs. Höfer. Michael Scheppler always encouraged me to focus on the big idea of configurable logic arrays and has been constantly supportive as I explored sometimes unconventional points of view. With his committed and constructive feedback he gave me important inspirations for finding progressive solutions of my work.

I also want to thank my second advisor, Prof. Dr.-Ing. Ulf Schlichtmann from the Technical University of Munich, for his friendly support and advice.

The work of the Master's Thesis students Anis Rahman, Walid Haj Taieb, Xianfeng Jiang and Bumei Zhai is well recognized. The internship students Thomas Delamare, Alexander Boulenger, Hadi Elias Maalouf and Avijit Saha did an excellent contribution.

I express my sincere thanks to the managers Dr. Christoph Heer, Steffen Buch and Holger Gryska for kindly supporting me to accomplish this thesis at Infineon Technologies AG, Munich, Germany.

Thanks to Prof. Dr.-Ing. Klaus Buchenrieder and Robert Fischer from the Bundeswehr University of Munich for power consumption on FPGAs discussions.

Thanks to Dr. Bingfeng Mei and Will Moffat for technical and personal help when I was visiting the IMEC labs in Leuven, Belgium.

A special thanks to Tobias Oppold from the University of Tübingen for reconfigurable architectures discussions.

Many thanks also go to my former colleagues at Infineon Technologies AG and my colleagues at Qimonda Flash GmbH.

I am most indebted to Jordi Carrabina from Autonomia University of Barcelona for introducing me to FPGAs.

I am grateful to James Ghirlando for proof-reading this manuscript.

I would like to thank all my friends. Thanks to my parents and the rest of my family for their endless support and love.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	This Thesis	4
1.3	Dissertation Organization	5
2	Related Work	7
2.1	Introduction	7
2.2	LUT-based FPGAs	7
2.2.1	Overview of Existing Solutions	7
2.2.2	Xilinx Virtex-II Pro	10
2.2.3	Xilinx FPGA Design Flow	13
2.3	LUT-based MPGAs	16
2.3.1	Overview of Existing Solutions	16
2.3.2	Altera HardCopy	20
2.3.3	Hardcopy Design Flow	21
2.4	Summary	22
3	Proposed LUT-based MPGAs	25
3.1	Introduction	25
3.2	r-MPGA	26
3.2.1	Architecture	26
3.2.2	r-MPGA Layout	27

3.3	Zelix MPGA	30
3.3.1	Architecture	30
3.3.2	Interconnect Representation	33
3.3.3	Clock Network and Power Grid	36
3.3.4	Zelix MPGA Layout	37
3.3.5	Circuit-Level Simulations	38
3.3.6	Discussion: Zelix MPGA Test	41
3.4	r-MPGA vs. Zelix MPGA	42
3.5	Zelix MPGA vs. Virtex-II Pro FPGA: Area	44
3.6	Discussion: Shift Register, and SRAM in the Zelix-LUT	45
3.7	Summary: Area	47
4	Zelix MPGA Conversion Design Flow	49
4.1	Introduction	49
4.2	Timing Problems in the Conversion Flow	50
4.3	Zelix MPGA Design Flow	54
4.3.1	Netlist Modifications	54
4.3.2	Zelix MPGA Placement	55
4.3.3	Zelix MPGA Routing	57
4.3.4	Place and Route Visualization	66
4.3.5	Buffer Insertion	67
4.3.6	Verification	75
4.3.7	Layout Creation	79
4.4	Discussion: Data Arrival Time Slack	80
4.5	Discussion: Adding Metallization	82
4.6	Summary	84
5	Sample Design Mappings on Zelix MPGA	87
5.1	Introduction	87
5.2	Experimental Preliminaries	88

5.3	Synthesis	89
5.4	Virtex-II Pro Placement and Routing	91
5.5	Zelix MPGA Routing	93
5.6	Zelix MPGA Buffer Insertion	96
5.7	STA Comparison	97
5.8	Zelix MPGA: Hold Time	100
5.9	Zelix Mask-Programming	101
5.10	Conversion Flow Time	102
5.11	Summary: Delay	102
6	Zelix MPGA Power Consumption Flow	109
6.1	Introduction	109
6.2	Power Consumption in CMOS Circuits	110
6.3	Power Estimation on Virtex-II Pro FPGA	111
6.4	Zelix MPGA Power Estimation Methodology	112
6.5	Power Consumption Estimation Flow Validation	114
6.6	Power Consumption Flow Results	115
6.6.1	Dynamic Power Consumption	116
6.6.2	Static Power Consumption	118
6.7	Summary: Power Consumption	118
7	Conclusion	121
7.1	Future Work	125
	Bibliography	127

List of Abbreviations

AND2	Two-input AND Gate
ALU	Arithmetic Logic Unit
ASIC	Application-Specific Integrated Circuit
ASSP	Application-Specific Standard Product
ATPG	Automatic Test Pattern Generation
BEST	Behavior Extracting Synthesis Technology
CAD	Computer Aided Design
CLB	Configurable Logic Block
CMOS	Complementary MOS
CPU	Central Processing Unit
DAT	Data Arrival Time
DCM	Digital Clock Manager
DSP	Digital Signal Processor
DSPF	Detailed Standard Parasitic Format
EDA	Electronic Design Automation
FF	Flip-Flop
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
HDL	Hardware Description Language
IC	Integrated Circuit

IEEE	Institute of Electrical and Electronics Engineers
IIR	Infinite Impulse Response
I/O	Input/Output
IOB	I/O Block
IP	Intellectual Property
ISE	Integrated Software Environment
ITC'99	International Test Conference 1999
JTAG	Joint Test Action Group
LUT	Look-up Table
MOS	Metal-Oxide-Silicon
MPGA	Mask-Programmable Gate Array
MUX	Multiplexor
NA	No Available
NMOS	N-channel MOS
NRE	Non-Recurring Engineering
OR2	Two-input OR Gate
PCB	Printed Circuit Board
PIP	Programmable Interconnect Point
PLA	Programmable Logic Array
PLL	Phase-Locked Loop
PMOS	P-channel MOS
RAT	Required Arrival Time
RC	Resistor-Capacitor
RTL	Register-Transfer Level
SPEF	Standard Parasitic Exchange Format
SRAM	Static Random Access Memory
STA	Static Timing Analysis
TCL	Tool Control Language

V_{DD}	Positive Supply Voltage
VHDL	Very High Speed Integrated Circuits HDL
VLSI	Very Large Scale Integration
VPGA	Via Programmable Gate Array
V_{SS}	Negative Supply Voltage, Ground
XDL	Xilinx Description Language
XOR2	Two-input XOR Gate
XST	Xilinx Synthesis Tool

List of Figures

1.1	Qualitative logic positioning in the 130-nm CMOS technology. The area does not include embedded blocks. Power consumption refers only to the dynamic contribution. Numbers estimated with [65, 108].	3
2.1	FPGA architecture (Xilinx Virtex-II Pro)	8
2.2	Xilinx Virtex-II Pro half slice [104]	11
2.3	Xilinx Virtex-II Pro double line (adapted from [64])	13
2.4	Xilinx Virtex-II Pro FPGA die photo [83]	14
2.5	Xilinx Virtex-II Pro FPGA detailed die photo	15
2.6	Xilinx Virtex-II Pro FPGA design flow	17
2.7	Altera Stratix Hardcopy architecture with an example of routing.	22
2.8	HardCopy Stratix migration flow	23
3.1	r-MPGA: array	27
3.2	Area optimization using a full-custom methodology in the r-MPGA [64]	29
3.3	r-MPGA: CLB layout	30
3.4	r-MPGA: tile layout	31
3.5	Zelix MPGA: array	32
3.6	Zelix MPGA: tile	33
3.7	Stick diagram of a tile	34
3.8	Three dimensional representation of a two-pin net	34

3.9	Three dimensional representation of a two-pin net with buffer inserted	35
3.10	Zelix MPGA clock network	36
3.11	Zelix LUT: schematic	38
3.12	Zelix LUT: layout	39
3.13	Zelix MPGA: CLB layout	40
3.14	Zelix MPGA: array of tiles layout	41
3.15	Delay of the Zelix 4-in LUT with different load capacitances . .	42
3.16	LUT with shift register	46
3.17	Memory cell for shift register	46
3.18	LUT with shift register, and SRAM memory functionality . . .	46
3.19	Memory cell for shift register, and SRAM memory	47
4.1	Data arrival time points. Adapted from [79], page 1.11	51
4.2	Hold time analysis. Adapted from [4], page 22-11	52
4.3	Zelix MPGA design flow	55
4.4	Zelix MPGA routing-resource graph	58
4.5	Maze router description	61
4.6	Extensions of the Lee Maze Router	62
4.7	MPGA viewer: placement	67
4.8	MPGA viewer: routing	68
4.9	An example of an RC interconnect	71
4.10	Tree representation of a net. S_0 is the source. $v_1, v_2, v_3, v_4, v_5, v_6$ are possible buffer positions. The downstream capacitance at v_2 is $C_1 + C_2 + C_3 + C_{12}$	72
4.11	Van Ginneken's operations	74
4.12	Example of the interconnect model	77
4.13	Three tiles layout.	80
4.14	Short path net in Zelix MPGA	81

4.15	Long path net in Zelix MPGA	82
4.16	Adding metal five correctly	84
4.17	Blockage problem	85
4.18	Adding metal six to solve the blockage problem	85
5.1	STA flow used in the experiments	89
5.2	Linear buffer delay.	98
5.3	Total area	103
5.4	Average delay without buffer insertion	104
5.5	Average delay with buffer insertion	104
5.6	Critical path delay without buffer insertion	105
5.7	Critical path delay with buffer insertion	105
5.8	Hold time slack of the ITC_b11 circuit	106
5.9	Layout of the FIR circuit	106
5.10	Detail of the FIR circuit layout	107
6.1	Power estimation flow	113
6.2	Comparison between transistor-level simulation (NanoSim, and Nanosim+PoweTheater.load), and proposed power consump- tion estimation methodology (PowerTheater).	115
6.3	Average dynamic power dissipation distribution.	117
6.4	Circuit ITC_b05 histogram of static probability.The number in brackets indicates the range of probability.	118
6.5	Circuit OC_fir histogram of static probability. The number in brackets indicates the range of probability.	119
6.6	Dynamic power dependence with the clock frequency of the OC_fir circuit	120

List of Tables

2.1	Characteristics of two Xilinx Virtex-II Pro devices	10
2.2	Area estimation of Virtex-II Pro blocks	16
2.3	MPGA metalization	20
3.1	r-MPGA and Zelix MPGA metalization	29
3.2	Tile area summary	30
3.3	Power consumption of the Zelix look-up table	40
3.4	One-length wire delay, and capacitance	43
3.5	Area comparison with XC2VP2 architecture (30 tracks per channel).	45
3.6	Area comparison with XC2VP125 architecture (30 tracks per channel).	45
3.7	Area comparison with XC2VP2 architecture (with shift register, and SRAM in the Zelix-LUT; 30 tracks per channel).	47
3.8	Area comparison with XC2VP125 architecture (with shift register, and SRAM in the Zelix-LUT; 30 tracks per channel).	48
4.1	MPGA netlist elements	56
4.2	Routing-resource graph parameters	59
4.3	MPGA base cost (b_n) values.	64
4.4	Cost function parameters	65
4.5	Simulation of three nets	78

5.1	Benchmarks circuits descriptions	90
5.2	Characteristics of the test examples	91
5.3	Net characteristics of the test examples	92
5.4	Number of tiles, local nets, and global nets after packing/ placement	93
5.5	Number of tracks per channels, and area. Plcmnt Effrt: Placement Effort, Rtng Effrt: Routing Effort, #trcks: number of tracks per channel, Reduc.: Reduction.	95
5.6	Number of tracks per channels, and area CONT'	96
5.7	Inserted buffers. Plc: placement effort, Rt: routing effort, HighArea: High routing effort with restricted area.	99
5.8	Buffers in the clock network. Low: Low placement effort, High: High placement effort.	100
5.9	Total number of buffers. Plc: placement effort, Rt: routing effort, HighArea: High routing effort with restricted area.	101
5.10	Data paths with larger delay than Virtex-II Pro	102
5.11	Conversion flow time (seconds)	107
6.1	Dynamic power consumption. Interconnect (Inter.) includes the buffers power dissipation.	117
6.2	Static power consumption on Zelix MPGA	120
7.1	Comparison between Zelix MPGA, and Xilinx Virtex-II Pro FPGA, with Altera Stratix HardCopy MPGA as a reference [3]. Percentage indicates reduction with respect to an FPGA.	124
7.2	Conversion time, and number of masks. Altera numbers from [16].	124

Chapter 1

Introduction

1.1 Motivation

This thesis deals with the question of versatile hardware chips which can be easily, and quickly configured for an application. An impressive innovation which emerged during the past two decades is field-programmable gate-arrays (FPGAs) technology. The major advantage of FPGAs is their ability to be programmed at the customer site, which makes them suitable for new ways of fast, and early prototyping.

One outstanding feature of FPGAs is that their scalable architecture is designed to support wide areas of applications. The hardware is supported by very powerful software tools which allow customers with little experience to design hardware on a low budget. Of course this means that technical support is required. Low up-front investments, in-the-field programmability and off-the-shelf ordering represent the highest flexibility for system development. FPGAs threaten application-specific integrated circuit (ASIC) business, not only because of the seamless transition from prototype to shipment quality, but also in aspects like product life time. The interesting question is whether FPGAs can be seen as a more versatile realization of the software defined system approach of processors. Can FPGAs better fulfil the promise of versatile hardware which is currently implemented using embedded software? From a

system level perspective, input/output (I/O) flexibility and parallel processing capabilities speak in favor of FPGAs. On the other hand, complex algorithms which allow periodic hardware re-usage speak in favor of processor-based architectures. However, the two above scenarios are typically not as clear in real applications. Other aspects like power consumption and time-to-market play an important role. As long as the battery technique for mobile applications makes no fundamental progress, power consumption remains a critical issue. The meaning of time-to-market is changing for two reasons: first, the life cycles of end products get shorter, and secondly, the complexity of the system increases. In other words, this means that systems need to be developed and verified in a shorter time at lower costs.

In the past ASICs seemed to be an answer. Due to a standardized design methodology and re-use of intellectual property (IP) (in the form of libraries, hard macros or soft macros), rapid design cycles where possible, while immediate silicon success was very likely. However, increased system complexity and more difficulties in physical verification are new challenges. The increased non-recurring engineering costs (NRE) make the decision in favor of an ASIC difficult. Higher minimum volume needs to be committed to redistribute the investment. Current alternative are application-specific standard products (ASSPs). These are ASICs for standardized market segments, usable by a multitude of customer. ASSPs include embedded system knowledge in the form of hardware, and include a defined degree of configurability and programmability.

FPGAs can be considered as a further step in this direction. Here the portion of embedded system know-how is lower, but the production know-how is increased, and the development methodology is better automated. On the other hand, FPGAs are cost prohibitive for mid-volumes (100k to 3 million units), and impracticable for large-volumes production (greater than 3 millions) [68].

Figure 1.1: Qualitative logic positioning in the 130-nm CMOS technology. The area does not include embedded blocks. Power consumption refers only to the dynamic contribution. Numbers estimated with [65, 108].

As a consequence of cost reduction, turn-around time, and low power dissipation, a new solution is necessary. Mask-programmable gate arrays (MPGAs) are an ideal candidate to fill the gap between FPGAs, and ASICs. In MPGAs, the logic mask-layers of a device are pre-defined, and pre-fabricated. The application specific design is achieved by creating custom metal layers. MPGAs may become important because they offer increased flexibility at the moderate cost of non-recurring engineering (e.g. mask cost) [61]. Figure 1.1 shows the logic positioning of FPGAs, MPGAs, and ASICs.

In ultra-deep submicron processes, technology aspects (e.g. process variations) strongly affect physical design and substantially increase the effort for verification, and yield optimization. On the other hand, system level design is absorbed with the task of increasing system complexity. A method of bridging this growing gap is to hide the technological aspects of a device by means of predefined structures, so the silicon design complexity is low. In this direction, FPGAs, and MPGAs have regular predefined structures.

A methodology to design digital integrated circuits (ICs) includes prototyping the design in an FPGA framework, and then converting the design to an MPGA for reasons of power reduction, and reduced cost per unit. Existing conversions use a target architecture with elementary cell structures, because this offers high density. The downside of this approach is the need for re-synthesis, and complete place and route. Due to the physical difficulties, a significant risk of corrupt timing integrity is taken.

For cost reasons, it is necessary to find an MPGA architecture that allows an easy conversion from an FPGA with a minimal effort. This architecture has to exploit layout regularity to minimize verification effort.

1.2 This Thesis

This thesis provides an automated conversion flow from an FPGA to an MPGA for a fast turnaround time.

For the thesis studies I use a commercial FPGA as a paradigm: the Xilinx Virtex-II Pro [99]. This industrial FPGA has a look-up table-based (LUT-based) logic cell, rich routing resources, and an embedded processor core. One of the reasons for using the Virtex-II Pro is the Xilinx design flow. The tools in the Xilinx flow are flexible and transparent to the user, i.e. it is possible to exchange one tool for another. Although I focus on a particular case, I believe that the results can be easily generalized.

The MPGA target architecture preserves the floor planning of the FPGA, but uses a different architecture for the routing resources. For this thesis the following assumptions are made: for a look-up table, the configurable latches are replaced by fixed connections to logical 1 and 0, respectively. Block static random access memories (SRAMs), and multipliers as well as embedded processor cores are not considered. The goal of the work is to show that the conversion flow is rather simple and high design security can be obtained.

The major contributions of this thesis include:

1. **r-MPGA:** A LUT-based MPGA that maintains the same configurable logic as a LUT-based FPGA. The interconnect is mask-programmed and maintains the interconnect segmentation of an FPGA.
2. **Zelix MPGA:** A novel MPGA architecture with a regular interconnect planning. The logic of an FPGA is maintained, but mask-programmed.
3. **Conversion Flow:** An automated conversion flow from an FPGA design to an MPGA design. The conversion flow includes:
 - (a) An MPGA router.
 - (b) A buffer insertion algorithm.
4. **Power Consumption Flow:** A methodology to estimate the power consumption in a LUT-based MPGA.
5. **FPGA vs. MPGA:** A concise comparison in terms of area, delay, and power consumption between an FPGA and an MPGA.

1.3 Dissertation Organization

This thesis is organized as follows: Chapter 2 reviews the work related to this thesis, including the architecture and computer aided design (CAD) flow of a commercial FPGA. The interconnect architecture of an FPGA is outlined in detail. The chapter also describes a commercial LUT-based MPGA. In Chapter 3 two new LUT-based MPGAs are presented. The first MPGA is similar to an FPGA. The second MPGA has a novel interconnect scheme. Chapter 4 presents an automated conversion flow. The flow has a novel MPGA router. Chapter 5 presents the experimental results within the conversion flow. A comparison between an FPGA and MPGA is discussed. A methodology to

estimate power consumption in FPGAs and MPGAs is presented in Chapter 6. Finally, Chapter 7 provides concluding remarks and directions for future work.

Chapter 2

Related Work

2.1 Introduction

This chapter reviews the architecture and CAD concepts of LUT-based FPGAs and LUT-based MPGAs. Section 2.2 provides some necessary background on FPGAs. The Xilinx Virtex-II Pro FPGA case is discussed in detail. Section 2.3 presents the concept of MPGAs. Specifically, the Altera Stratix HardCopy MPGA is described. Section 2.4 summarize the chapter.

2.2 LUT-based FPGAs

2.2.1 Overview of Existing Solutions

The most common reconfigurable devices today are FPGAs. FPGAs can be programmed by the user in minutes. FPGAs can be used to implement any digital hardware design. These are devices used both as prototyping platforms and as reconfigurable alternatives to ASICs. FPGAs can achieve much higher levels of integration than programmable logic arrays (PLAs), due to their more complex routing architectures, and logic implementations. These configurable devices also offer features such as built-in hardwired processors, substantial amounts of SRAM memory blocks, clock management systems, and support for

Figure 2.1: FPGA architecture (Xilinx Virtex-II Pro)

many of the latest, very fast device-to-device board-level signaling technologies. FPGAs are used in a wide variety of applications, ranging from data processing, and storage, to instrumentation, network communications, and digital signal processing.

Figure 2.1 shows a conceptual diagram of a typical FPGA. There are three key parts of its structure:

1. I/O blocks (IOB)
2. Hard macros: SRAM block, multiplier, and digital clock manager (DCM)
3. Logic blocks with the configurable interconnect (tile)

Typically, the I/O blocks are arranged as a ring around the outer side of the FPGA die. There are two different types of I/O blocks: global I/O blocks, and programmable I/O blocks. The global I/O blocks include dedicated units for configuration, JTAG test¹, clock, and power/ground connection. A programmable I/O block provides individually programmable input, output, or bi-directional (any combination of the input, and output configuration) access to one of the I/O pins on the exterior of the FPGA package.

Hard macros are used to improve the area and delay efficiency of the FPGA. Common blocks include SRAMs, multipliers, transceivers, processors, and DCMs. A generic FPGA consists of numerous configurable logic blocks; each of them having the capability to implement a wide range of digital logic functions. A Boolean logic function can be obtained thanks to a LUT-based or multiplexer based (MUX-based) hardware implementation. The configurable interconnect is used to connect the input, and output of the logic blocks. It is designed to support any netlist mapping without major routing congestion. Typically the array is two-dimensional, although there are some solutions which are based on a one-dimensional architecture (i.e. a row-based architecture). The latter however do not take advantage of having short connections to next neighbors in the vertical, and horizontal direction.

In a so-called island style FPGA, the logic blocks are arranged in a two-dimensional array with horizontal routing channels between rows of blocks and vertical routing channels between columns. Each routing channel comprises a bundle of routing tracks for signal transport. At each crossing point of routing channels, there is a configurable switch-matrix, which allows change of direction or communication with its neighboring logic block. Thus, a so-called tile comprises a logic block, a switch matrix, and routing channels. By pure abutment of tiles the array size can be varied (i.e. no additional routing resources need to be provided after the tiles are placed.)

¹In some FPGA devices, the JTAG port is also used for configuration.

Device	Transceiver Blocks	Processor Blocks	18Kb SRAM Blocks	18x18 Mult. Blocks	DCM	Max. User I/O Pads
XC2VP2	4	0	12	12	4	204
XC2VP100	0, 20 or 24	4	556	556	12	1200

Table 2.1: Characteristics of two Xilinx Virtex-II Pro devices

The configuration of the FPGA is done by means of programmable switches. There are different ways of implementing a programmable switch. Programmable switches that are currently in use in commercial FPGAs are SRAM cells, and Flash memory cells. The FPGA market is coped with SRAM cells because this is a cost-effective technology process.

There are two major players in the FPGA market: Xilinx Inc. [102], and Altera Inc [8]. Both use architectures which are array-based, and have logic clusters at the lower hierarchy level. They have two families, one for high performance, and high complexity, and another one for low cost, and higher volume. Both companies follow the trend of embedding more, and more optimized macros into the array (e.g. digital signal processors (DSPs), transceivers, ...).

2.2.2 Xilinx Virtex-II Pro

The studies of this thesis use the Xilinx Virtex-II Pro [99] device. The Xilinx Virtex-II Pro is a 130nm CMOS nine-layer (copper) FPGA device. The Xilinx Virtex-II Pro has embedded blocks. Table 2.1 shows the different block number of two devices. The smallest device, XC2VP2, has no embedded PowerPC processor blocks. It consist of 352 logic tiles. The largest device, XC2VP125, has 13900 logic tiles.

Virtex-II Pro has an island style architecture. It consists of a two-dimensional array of configurable logic blocks (CLBs), and programmable interconnect resources. Each CLB has four identical sub-blocks, called slices. Figure 2.2 shows the components of a half slice. A slice consist of two four-input LUTs,

Figure 2.2: Xilinx Virtex-II Pro half slice [104]

two Flip Flops, gates (two AND2, one OR2, two XOR2), multiplexors, inverters, and buffers. The LUTs are used to map four-input Boolean logic. The gates are used to implement arithmetic functions such as carry chains. Four multiplexors are used as mapped multiplexors (after synthesis), and the other multiplexors as configurable multiplexors (i.e. to do programmable routing inside the slice). The inverters, and the buffers are needed to implement the different clock types allowed in the FPGA. The four slices of one CLB share a common programmable I/O crossbar for communicating with a switch matrix. The programmable I/O crossbar is also used for local signal communication between slices. The switch matrix realizes communication with others CLBs. The switch matrix can route a signal north, south, east or west. The topology of the switch matrix in the Virtex-II Pro is a special disjoint type [40]. A CLB, together with its associated switch matrix, is called tile.

Routing Resources

The programmable interconnect is responsible for connecting the inputs of the logic cells, and the outputs. The Virtex-II Pro has two hierarchical levels: local interconnect, and global interconnect. The local interconnect is the programmable I/O crossbar mentioned before. The global interconnect consist of vertical, and horizontal routing channels. A wire in a channel can span two tiles (double lines), six tiles (hex lines), or all the row/column (long lines). The Virtex-II Pro has 40 double lines, 120 hex lines, and 24 long lines in a channel. All these lines are routed within the switch matrix. Figure 2.3 shows the staggered pattern of a double line. A special type of wire can connect pins of two CLBs without using the switch matrix: direct neighbor lines. The direct neighbor lines connect adjacent CLBs. This line goes directly from an output crossbar to an input crossbar.

The number of lines that can be connected from one CLB to another CLB in one direction (east, west, north or south) is called "number of tracks per channel".

Area Estimation

The die photo of the Xilinx Virtex-II Pro XC2VP7 device is shown in Figure 2.4. The die photo is taken from the presentation of [62]. No area for the photo die is given. Xilinx Virtex-II Pro has memory blocks of 18Kb. A detail of the die photo is shown in Figure 2.5. I calculate the area of a tile using the area of a memory block. The area of four memory blocks is equal to 24 tiles. An internal Infineon tool [38] gives us the area of an 18Kb dual-port embedded memory block. Therefore the area of a tile is $206\mu\text{m} \times 145\mu\text{m}$. In Table 2.2, the area for the others blocks (except embedded PowerPC processor) is reported.

Figure 2.3: Xilinx Virtex-II Pro double line (adapted from [64])

2.2.3 Xilinx FPGA Design Flow

A typical design flow for FPGAs consists of a concatenated set of CAD tools. The Virtex-II Pro device is supported by a complete set of Xilinx integrated software environment (ISE) development tools [101], with additional tools available from a variety of third party companies. Figure 2.6 presents the general overall Virtex-II Pro design flow. The standard design flow for Virtex-II Pro FPGAs consist of three major steps: design entry, and synthesis, design implementation, and design verification.

Figure 2.4: Xilinx Virtex-II Pro FPGA die photo [83]

The start of the design flow is a schematic entry or a high-level description of the hardware, expressed in high-level hardware description languages (HDL) such as VHDL or Verilog. The description is read by a synthesis program, which maps the HDL into a network of Boolean equations, flip-flops, and pre-defined modules. During the synthesis process, the Boolean equations are optimized with respect to estimated implementation area, and delay. The optimizations performed at this stage are limited to those that can benefit circuit implementations on any medium, not just FPGAs. The Boolean equations are then first mapped into a circuit of FPGA LUTs through the technology mapping process. During the technology mapping, the circuit is again optimized with respect to the estimated implementation area, and delay. The optimizations are targeted towards specific implementation technologies. Area is typically optimized by minimizing the number of LUTs or logic blocks that are required to implement the circuit. Delay is often optimized by minimizing

Figure 2.5: Xilinx Virtex-II Pro FPGA detailed die photo

the number of LUTs or logic blocks that are on the estimated critical paths of the circuit. Xilinx offers a free available synthesis tool called XST [100]. An example of a third party synthesis tool is the Synplify-Pro from Synplicity [82]. The design implementation steps consist of packing, placement, routing, and bitstream generation. The packing process groups logic blocks into slices, each of which usually contains several LUTs, and flip-flops. The specific location of each slice on the target FPGA is determined during the placement process. A placement program assigns each slice to a unique location to optimize delay, and minimize wiring demand. During the routing process, a routing program is used to connect logic blocks by determining the configuration of the programmable routing resources. The main task of all routing programs is to successfully establish all connections in a circuit using the limited amount of physical resources available on a target FPGA. The other task of the routing programs is to minimize delay by allocating fast physical connections for critical paths. The synthesis, and the technology mapping are together commonly

Resource	Area (mm^2)
Tile	0.03
SRAM	0.18
Multiplier	0.06
DCM	0.29
I/O PAD	0.01

Table 2.2: Area estimation of Virtex-II Pro blocks

called the front end of the FPGA CAD flow. The packing process, placement, and routing steps are commonly called the back end of the FPGA CAD flow. Finally, from the design, and placement and routing information, a bitstream is created for subsequent programming of the Virtex-II Pro device.

Design verification is the process of testing the function, and performance of a design. The verification can be realized with functional simulation, static timing analysis, and in-circuit verification. Functional simulation determines whether the logic of the designed circuit is correct. Functional simulation can take place in early stages of the design (e.g. in the HDL phase). The static timing analysis verifies that the design meets the timings specifications. In-circuit verification tests the circuit under typical operating conditions.

2.3 LUT-based MPGAs

2.3.1 Overview of Existing Solutions

The routing resources in an FPGA consume about 88% of the total area [45]. To reduce this area penalty, FPGA vendors use a high number of metal layers,

Figure 2.6: Xilinx Virtex-II Pro FPGA design flow

and the latest process technology in their device design. The use of MP-GAs is addressed to reduce the high penalty area of FPGAs, exchanging user-programmability for fab-programmability. In an MPGA, all the mask layers that define the circuit device are pre-defined, except those that specify the final metal layers. These metal layers are customized to connect the logic gates, thereby implementing the desired circuit. The programmable interconnect resources of an FPGA are replaced with custom metal layers, and vias.

Like FPGAs, MPGAs are based on a fundamental element called tile. This tile contains a small amount of generic logic. The generic logic can be CMOS transistors, standard cells or LUTs. The connections between the tiles are done with custom masks (mask-programmed). In some cases the LUTs are configured with custom masks. A special case is to program the MPGAs

only with via masks instead of metal, and via masks. It is possible to distinguish between three classes of MPGAs, depending on the complexity of the basic component-cell: CMOS-based MPGAs [34, 53], cell-based MPGAs (also called structured ASICs) [74, 54, 1], and Look-up Table based (LUT-based) MPGAs [3, 46]. There have been CMOS-based MPGA products under the name of Gate Arrays since the 80's. CMOS-based MPGAs are in disuse because the poor quality of their CAD tools cannot deal with the complexity of current designs. Moreover, the small market niche of CMOS-based MPGAs has been taken by modern FPGAs. In recent years, companies that offer Cell-based MPGAs (under the name of structured ASIC) have emerged. A Cell-based MPGA consist of an array of blocks. Each block contains a set of 10 to 15 basic cells (AND gates, OR gates, FFs, ...). As opposed to CMOS-based MPGAs, Cell-based MPGAs typically contain additional prefabricated elements such as gigabit transceivers. In comparison to Cell-based MPGAs, few companies offer LUT-based MPGAs. This is because LUT-based MPGAs require more specialised CAD tools than those for LUT-based FPGAs. Home-based development of CAD tools by LUT-based FPGA companies, means that no third companies offer tools for such placement. A boom of Cell-based MPGAs or LUT-based MPGAs has been seen in industry. In academic circles, we can find studies which focus on Cell-based MPGA synthesis [36] or LUT-based MPGAs architecture exploration [61].

There is one case that can be compared with the result of this thesis: Altera Hardcopy [3]. Altera Hardcopy is a LUT-based MPGA and, as in the case of this work, a design mapped to an FPGA can be migrated without re-synthesis. In Section 2.3.2 the concept of this architecture is explained.

A comparison between MPGAs includes aspects of delay, power consumption, and cost. While waiting for a public fair comparison, I assume here general concepts. Delay, power consumption, and area are small in Cell-based MPGAs. This is a result of the small granularity of a Cell-based MPGA.

Synthesis, and placement can be executed in a more efficient way, reducing the area, and therefore the delay, and power consumption. The cost of design is low in the LUT-based MPGA. Assuming that the design has been prototyped in an FPGA (as is the usual case), the migration to an MPGA can take about 14 days in a LUT-based MPGA [16], and more than 2 weeks in a Cell-based MPGA [63]. This difference is due basically to the timing closure problems associated with the re-synthesis, and re-placement. Also important is the metalization of the MPGAs. Table 2.3 shows the programmable, and prefabricated mask layers of commercial MPGAs. We can observe that the metal stack is of six to ten layers. The via programmed architectures need special process technology features such as e-beam lithography [35]. The number of pre-fabricated metal layers for the NEC architecture is unknown. The eASIC architecture needs an external memory device to configure the LUTs. The Altera Hardcopy uses one metal layer to configure the LUTs. The mask set cost for a cheap 130nm CMOS technology with 6 metal layers is around half a million euros. The cost of two metal layers with three vias is around 0.1 million euros. This is motivation for the use of a low metal stack number for programming, and the re-use of the mask set.

There is an interesting case from Carnegie Mellon University: VPGA [86]. The VPGA is a LUT-based MPGA, programmed only with vias. The VPGA has an island-style architecture. After design exploration with a 130nm CMOS process, they found that a three-input LUT is the best choice [57]. Also, they claim that the better logic cell is a combination of basic logic cells (as are NAND gates) with LUTs [43]. Although a high effort has been spent on these investigations, there are some points that make the investigation unclear. In some experiments, they use a cell-based synthesis tool (Synopsys Design Compiler) [81]. As this tool is not an FPGA synthesis tool, it is easy to conclude that the use of NAND or XOR gates is the most efficient in terms of area. The

Vendor	Architecture	Fixed Metal Layers	Prog. Metal Layers	Prog. Vias	Total Metal Layers
LSI Logic [1]	Cell-based	5	5	4	10
NEC [54]	Cell-based	N/A	2	2	N/A
Fujitsu [33]	Cell-based	4	4	4	8
Lightspeed [48]	Cell-based	4	4	3	8
ViASIC [97]	Cell-based	8	0	1	8
Faraday [32]	LUT-based	2	4	3	6
eASIC [46]	LUT-based	8	0	1	8
Altera [3]	LUT-based	5	3	3	8

Table 2.3: MPGA metalization

experiments are done with four small circuits (e.g. an ALU). The architectural exploration uses a different flow adapted from academic tools [15], and no comparison with the FPGA is provided. Moreover, what is unknown in these studies is whether there is any real benefit in comparison with state-of-the-art FPGAs.

2.3.2 Altera HardCopy

Altera has three different LUT-based MPGA families. I discuss the HardCopy Stratix [3], as it is closer to this work. Hardcopy Stratix is a LUT-based MPGA with the same architecture as the Stratix FPGA. It is manufactured in 1.5V, 130nm eight layer all-copper metal fabrication process from the TSMC foundry. HardCopy Stratix devices are architecturally equivalent, and have the same features as the corresponding Stratix FPGA. HardCopy Stratix includes the same hard-blocks (memory, DSP blocks,...) as the Stratix FPGA. This MPGA has the same package as the corresponding Stratix FPGA. A circuit

that is prototyped in a Stratix FPGA can be migrated to a HardCopy Stratix without any additional change to the final pin-out of the PCB board.

The HardCopy Stratix consists of a two-dimensional row- and column-based architecture to implement custom logic. The logic array consist of tiles. A tile has 10 slices. Each slice contains carry chains, a four-input LUT, a register, an XOR gate, and multiplexers. The XOR gate is used for efficient implementation of arithmetic logic. The multiplexers are used to control the signal paths. The communication between tiles is fully customized with a semi-custom tool. In comparison with the Stratix FPGA, the HardCopy MPGA has an area reduction of up to 70%, an average of 50% performance improvement, and an average of 40% less power consumption [3].

2.3.3 Hardcopy Design Flow

Altera offers the customer its Quartus II software, for the design of HardCopy Stratix devices [5]. The final back-end is realized in-house. The migration flow is shown in Figure 2.8. The migration process uses Altera proprietary, and third-party EDA tools. The flow starts with the FPGA Verilog HDL structural netlist that describes how each structural element (such as LUT) is connected to another structural element. To optimize the run time of the placement tool, the placement information from the FPGA is used. Note that the HardCopy Stratix has a new placement, excluding the I/O blocks. In the routing stage, physical rule violations are fixed. Signals with high fan-outs are buffered. Antenna violations, crosstalk violations, and metal spacing violations are also fixed in this stage. The routing step uses a conventional ASIC routing tool [60]. The resulting irregular routing is shown in Figure 2.7. In the physical verification, timing is checked, and corrected after place and route. To achieve accurate timing simulations, resistances, and capacitances are extracted from the layout view. Setup time violations are corrected in

Figure 2.7: Altera Stratix Hardcopy architecture with an example of routing.

two ways. First, extra buffers can be inserted to speed up slow data signals. Secondly, if buffer insertion does not completely fix the setup violation, the placement can be re-optimized. Hold-time violations are fixed by inserting delay elements into fast data paths.

2.4 Summary

In this chapter, FPGAs, and MPGAs have been described. FPGAs are programmable devices at the customer site. Modern FPGAs have embedded hard

Figure 2.8: HardCopy Stratix migration flow

IPs such as memory blocks. FPGAs have a simple design flow. In comparison with other VLSI structures, FPGAs have a large area, and high power consumption. The Xilinx Virtex-II Pro was presented in detail. MPGAs are programmable devices at the fab site. In MPGAs, all the mask layers are pre-fabricated, except those reserved for customization. We studied the case in which the MPGA architecture is based on LUTs. The Altera Stratix HardCopy was studied in detail. The Stratix HardCopy has a conversion flow which excludes the need for re-synthesis. As a result of semi-custom routing, the physical verification is the same as for ASIC methodologies.

Chapter 3

Proposed LUT-based MPGAs

3.1 Introduction

I named the first LUT-based MPGA presented in this thesis r-MPGA¹, and the second Zelix MPGA². Both architectures have the same gate-level logic elements as the Xilinx Virtex-II Pro³. For the sake of simplification, the embedded hardware blocks (such as 18 Kb SRAM memory modules) are not implemented in the MPGAs.

Both MPGAs consist of two-dimensional arrays of Configurable Logic Blocks (CLBs), and mask-programmable interconnect resources. The CLB has four slices.

Current LUT-based MPGAs have an irregular routing structure (e.g. Altera HardCopy-II [3]). This is because final interconnections are customized by a metal masking process, and therefore it is possible to route with a semi-custom tool. The mask-programmable interconnect in our MPGAs is planned in the same fashion as an FPGA, i.e. we have a predefined regular interconnect structure. The benefits of this regular interconnect structure are the possibility to

¹r-MPGA means either "Regular MPGA" or "Rahman MPGA" (Mr. Rahman did the first layout studies of this architecture [64]).

²Zelix is an evolution of the Arabic word "Al Zelij" that means "Polish Stone". In Spain this word becomes Azulejos. Azulejos are the famous Spanish, and Portuguese regular tiles on a wall.

³This is not true for the LUT used as shift register. This case is discussed in Section 3.6.

use an FPGA routing tool on one hand, and easy-predictability of physical phenomena (e.g. delay, crosstalk) on the other hand [41]. The second simplifies the verification process.

The rest of this chapter is organized as follows: Section 3.2 presents the r-MPGA architecture, and layout. In Section 3.3, the Zelix MPGA is described. The r-MPGA, and the Zelix MPGA are compared in Section 3.4. A comparison between the Zelix MPGA, and the Virtex-II Pro FPGA is provided in Section 3.5. Section 3.6 discusses the addition of shift register, and SRAM functionality in the Zelix-LUT. Section 3.7 summarizes the chapter.

3.2 r-MPGA

The concept of the r-MPGA is to have an MPGA that preserves the structure of the Virtex-II FPGA. The difference is the way the routing, and logic is programmed. In an FPGA, the programming of the routing tracks, and the logic cells is done by using latches. In the r-MPGA, the programming is done by via, and mask customization. The later concept was introduced by Xilinx in its XC3000 family [56].

3.2.1 Architecture

Figure 3.1 depicts the array architecture of the r-MPGA. The CLB of the r-MPGA is exactly the same as that of the Virtex-II. The Virtex-II I/O programmable crossbar is maintained in the r-MPGA. To save active area, the I/O crossbar is not fully populated as in the FPGAs.

The CLB is programmed by applying V_{DD} or V_{SS} to the configuration signals. The hierarchical routing segments of the Virtex-II are maintained. For the switch matrix, a concept of via programmability has been developed. After routing of one design, via connections are provided between two perpendicular metal layers. In this sense, the r-MPGA can be routed with the route

Figure 3.1: r-MPGA: array

information of the Virtex-II. We can note that the only difference between this architecture, and the Virtex-II is the programming method. The r-MPGA is via programmed, while the Virtex-II is configured with latch cells. As we can see in Figure 3.1, the segmented interconnects always bypass the switch matrix, i.e. as a concept, the logic of a CLB can access all the wires of its tile. Compared with an FPGA, the r-MPGA is more robust (i.e. protected to soft-errors).

3.2.2 r-MPGA Layout

The technology used in the r-MPGA design is an Infineon 130nm six copper metal layers CMOS process. I use a full-custom methodology with hand-crafted cell [25]. The VLSI layout takes advantage of the intrinsic hierarchical regularity of the MPGA architecture, i.e. first the basic logic (LUTs, FFs, ANDs, etc.) is designed, then the logic is resembled in slices. A set of slices

is a CLB, a CLB with a switch matrix is a tile, and finally an array of tiles is the MPGA.

I base all the layout studies on a tile level. The final MPGA array consists of a distributed arrangement of this tile.

The design methodology consists of optimizing the basic cells to find an area optimal layout. In this sense, several versions of the CLB layout have been realized. The first version is based on a design involving hand placement of standard-cells. The resulting layout was much bigger than other commercial ones. Area optimization was necessary to have a comparable layout. Critical parts in the layout were identified for area improvement. For example, the 4-input LUT is implemented with 2:1 MUXs in the first version. In the new version the 2:1 MUXs are substituted by full-custom pass-gates. After four versions following this methodology, an optimal layout has been obtained (see Figure 3.2). It was shown in [64] that no improvement in terms of area can be achieved using this methodology. Figure 3.3 shows the layout of the CLB. The floorplan of the CLB is exactly the same as the Virtex-II Pro [103].

To implement the top-level routing, space has been left in the floorplan. As the hierarchical interconnect of the Virtex-II is maintained, we need to use three metal layers. The lines have a staggered pattern to span several tiles. Unlike the Virtex-II Pro wires, the r-MPGA wires can be accessed by all the tiles spanned by the wires. As an example, a common hex-line (a wire that spans six tiles) in the Virtex-II Pro can be accessed only by the two end tiles. In the r-MPGA a hex-line can be accessed by all the six tiles which it spans. The top-level routing can be buffered at the start, and end points. These buffers are located in the physical area reserved for the top-level routing.

The CLB is programmed by means of configurable inputs set to V_{DD} or V_{SS} . The switch matrix is via-programmed. Only via connections must be provided, but this requires a special technology with fusible vias. In practice the metalization between the vias is provided to avoid antennas. The resulting layout

Figure 3.2: Area optimization using a full-custom methodology in the r-MPGA [64]

	Cell Design	Inter-cell Connection	CLB Configuration	Top-level Routing	Power Grid
r-MPGA	M1	M2,M3	M3	M3,M4,M5	M6
Zelix MPGA	M1	M2	M3,M4	M3,M4	M4

Table 3.1: r-MPGA and Zelix MPGA metalization

uses two vias, and three metal layers for the mask-programming. The metal stack uses six metal layers. Table 3.1 shows the metalization planning. Metal 6 is used for the power grid. As the interconnect architecture is maintained, it is easy to import the routing from the FPGA flow. Figure 3.4 shows the final tile layout.

The total area of the tile is $373\mu\text{m} \times 465\mu\text{m}$. As we can see in Table 3.2, the major part of the area is dedicated to the global routing resources. About 11% of the total area is used by the I/O crossbar of the CLB.

Figure 3.3: r-MPGA: CLB layout

	Zelix MPGA	r-MPGA
Top-level Interconnect	57%	68%
Local Interconnect	0	11%
CLB Logic	43%	21%
Total	0.005mm ²	0.17mm ²

Table 3.2: Tile area summary

3.3 Zelix MPGA

The Zelix MPGA architecture has the same topology, and the same gate-level logic elements as the Xilinx Virtex-II Pro FPGA. For the sake of simplification, the embedded hardware blocks (such as 18 Kb SRAM memory modules) are currently not implemented in the MPGA.

3.3.1 Architecture

The Zelix MPGA consists of a two-dimensional array of tiles. Each tile has CLBs, and mask-programmable interconnect resources (see Zelix MPGA in Figure 3.5). The CLB has four slices. Each slice is made of four-input LUTs,

Figure 3.4: r-MPGA: tile layout

two flip-flops, simple logical gates (two AND2, one OR2, two XOR2), four multiplexors, inverters, and buffers. The LUTs are used to map Boolean logic functions of up to four inputs. The gates, and the multiplexors are used to implement special functions such as carry chains. The inverters, and the buffers can be used for the different clock types.

To solve the problems of the r-MPGA, I implemented a new interconnect scheme. Two different topologies have been studied: XC4000 style, and XC5000 style [84]. The XC5000 style is the topology used in the Virtex-II Pro architecture. In the XC4000 style the input/output signals of the CLB are directly connected to the tracks of one adjacent channel, and then distributed to the other tracks via a switch matrix. In the XC5000 style, the input/output signals of the CLB are connected to a switch matrix, and then the signals are distributed to the other tiles. We found a similar area in both styles. We observed a better CLB pin access, and a more flexible switch matrix in the XC5000 style, compared to the XC4000 style. Therefore, I used the XC5000 style. The

Figure 3.5: Zelix MPGA: array

Zelix MPGA has two hierarchical levels for the interconnect: top-level interconnect, and local interconnect. The local interconnect is a mask-configurable input/output crossbar (see input, and output matrix in Figure 3.6). Since the configuration is done with vias, we have a full-populated crossbar. A via between metal 2 and metal 3 has a small resistance of 0.9Ω in a 130nm CMOS technology (copper).

The top-level interconnect is used for signal communication between CLBs. This communication is done through a switch matrix. The topology of the switch matrix is a fully-populated crossbar. The Zelix MPGA has only one type of top-level wire: 1-segment line, i.e. one wire can only expand one tile. This segmentation simplifies the task of the routing tool, and the MPGA device design. Although the segmentation is a 1-segment line, there is no difference between two 1-segment lines, and a 2-segment line (with the same direction) in the final mask layout. A 1-segment line topology is also useful for buffer insertion planning. For instance, in a wire of length two, a 2-segment line has

Figure 3.6: Zelix MPGA: tile

only two points for inserting a buffer (input, output), but two 1-segment lines have three points (input, mid point, output). The current implementation of the Zelix MPGA has 30 tracks per channels.

3.3.2 Interconnect Representation

Stick diagrams can be used for interconnect visualization. Figure 3.7 shows the floorplan of a tile, and the stick diagrams of three nets. The floorplan is divided into the CLB area, and the top level interconnect area. Figure 3.8 shows a two pin net drawn in three dimensions. Figure 3.9 shows the same net with a buffer inserted.

An expression for the area of a tile can be extracted from the architectural parameters. Using the floorplan of a tile (see Figure 3.7, and the switch matrix

Figure 3.7: Stick diagram of a tile

Figure 3.8: Three dimensional representation of a two-pin net

Figure 3.9: Three dimensional representation of a two-pin net with buffer inserted

in Figure 3.6), we see that the area is:

$$\begin{aligned}
 Area_{tile} &= W * H + N_{trcY} * k * H \\
 &\quad + (2 * N_{trcX} + 2 * N_{trcY}) \\
 &\quad * (k * W + N_{trcY} * k^2)
 \end{aligned} \tag{3.1}$$

Here, W and H are the width, and height of a CLB respectively. The numbers of tracks per horizontal, and vertical channel are N_{trcX} and N_{trcY} respectively. k is the width of a wire, and the minimum distance between two wires. W and H are fixed after the physical layout of the CLB is done. k depends on the technology process used. N_{trcX} and N_{trcY} are architectural parameters, and vary when one circuit is mapped to another. The dependency on the number of tracks per channel mapping different circuits using an FPGA routing tool is discussed in [15]. In Chapter 5, the case for the MPGA is discussed.

Figure 3.10: Zelix MPGA clock network

3.3.3 Clock Network and Power Grid

Although it is possible to use the programmable interconnect for routing the clock, the Zelix MPGA has a dedicated global clock network. The main issues when designing the clock distribution are the clock skew, and the clock jitter. The clock skew is the time difference between two rising clock edges due to spatial separation. The clock jitter is the inaccuracy of the clock due to temporal separation. Clock jitter is caused by supply voltage noise, signal coupling noise, and the inherent PLL jitter. I assume that no clock jitter is present in the Zelix MPGA. The clock skew is investigated with transistor-level simulations in Section 3.3.5. As the Virtex-II Pro, the Zelix MPGA has a clock network with an H-tree topology. It has been shown that an H-tree topology is the best solution with respect to the clock skew, power consumption, and layout area [107]. Figure 3.10 shows the Zelix clock network. The Zelix MPGA is divided into four quadrants (NW, SW, NE, and SE). The root of the clock is always buffered. Each branch of the tree is also buffered. A branch has a depth of eight tiles. Each tile has four clock buffers for the four slices. The two flip-flops of a slice share one clock buffer.

The power grid can be implemented in two different ways. In the first, the

power grid is created in the same metalization level as the global interconnect. This option is reasonable in the sense that the top-level interconnect uses a small amount of the final area not used for routing. In case of power grid problems such as current drops [106], a dedicated power grid can be implemented.

3.3.4 Zelix MPGA Layout

Standard cells, and hand-crafted cells are used. For reasons of qualification the standard cells are used as much as possible. Only in the LUT was it necessary to use hand-crafted cells. In this case, I re-used the hand-crafted cells from the r-MPGA. Figure 3.11 shows the schematic view of the LUT. The layout is shown in Figure 3.12. Circuit simulations with Synopsys NanoSim [80] have been done to show that the Zelix logic has the same performance as Virtex-II [104]. Table 3.1 shows the metalization planning. Metal 4 is used for the power grid. Because the top-level segmentation routing is simple, it needs fewer metal layers than the r-MPGA does. The clock network, and the FFs control signals use the Metal 1, Metal 2, and Metal 3 layers. Figure 3.13 shows the layout of one CLB. We can note that there is no active area for the input/output crossbar. The input/output crossbar is situated over the CLB, and configured with the Metal 3, Metal 4, and Via34. The input/output CLB pins are arranged in the mid vertical line of the CLB. The two LUTs, and two FFs occupy practically all the area of a slice. Figure 3.14 shows a detail of an array of tiles. We can see that the one-segmentation interconnect fits into a more regular, and compact layout compared to the r-MPGA.

The total area for the tile layout is $48\mu\text{m} \times 112\mu\text{m}$. 57% of this area is dedicated to the routing resources. In contrast to the FPGA, and the r-MPGA, the area is equally distributed between the logic, and the interconnect.

Figure 3.11: Zelix LUT: schematic

3.3.5 Circuit-Level Simulations

Circuit-level simulations are needed to determine the delay and power consumption of a cell. As I am using standard cells in the Zelix MPGA, the logic elements have already been specified by the technology vendor. The only logic element with unknown delay and power is the LUT, so I have simulated it to find its delay and power consumption. Circuit-level simulations are also used to determine clock skew. Finally, simulation of various nets is essential in setting up an interconnect model. These interconnect simulations are presented in Chapter 4.

Zelix Look-up Table

The LUT was designed to have a delay within the range of the Virtex-II Pro, i.e. 280ps. Checking with different output capacitances (see Figure 3.15) shows that the LUT delay is longer than 280ps when the load capacitance is

Figure 3.12: Zelix LUT: layout

larger than 30fF. A NAND2 gate has a fan-in capacitance of approximately 4fF. Therefore, the LUT can have a fan-out of up to 8 NAND2 gates to stay within the range of the Virtex-II Pro.

I have analyzed the power consumption with and without parasitic extraction, using the Assura RCX tool, extracting capacitances as decoupled capacitances. I have investigated different cases for the study; the input vectors are set, in the sense that we have different frequencies at the output of the LUT. I have set one case with no activity at the output of the LUT; the maximum output toggling signal is 100MHz, the supply voltage is 1.5V, and the load capacitance of the LUT is 14.5fF. Table 3.3 gives the simulation results. The circuit with the parasitic annotation consumes more power than that without, due to having more capacitors in the extracted view, which leads to a higher power consumption. The static power consumption is 6nW. The power consumption of the Zelix LUT is $0.351\mu\text{W}/\text{MHz}$ without extraction, and $0.505\mu\text{W}/\text{MHz}$ with extraction. The LUT of the Stratix HardCopy has a power consumption around $0.595\mu\text{W}/\text{MHz}$, as calculated using the Altera power estimation

Figure 3.13: Zelix MPGA: CLB layout

spreadsheet [6]. The LUT of the Xilinx Virtex-II Pro FPGA has a power consumption of $0.732\mu\text{W}/\text{MHz}$, found by mapping a set of circuits with the Xpower tool. This value was set in [40], showing good accuracy.

Clock Network

The clock network was simulated to find the worst skew case, with the capacitances of the clock tree extracted. According to the simulation, the clock

Parasitics	Output toggling signal			
	100MHz	50MHz	25MHz	0MHz
without (μW)	21.90	15.15	13.30	12.31
with (μW)	31.35	21.90	19.05	18.45

Table 3.3: Power consumption of the Zelix look-up table

Figure 3.14: Zelix MPGA: array of tiles layout

skew is 130ps. A large skew can violate the hold time. In Chapter 4, I discuss hold-time violations.

3.3.6 Discussion: Zelix MPGA Test

Although the silicon test was not taken into account in the first Zelix MPGA, I discuss in this section the ways it can affect the Zelix MPGA design. A VLSI chip has to be tested to ensure that it was manufactured correctly, the type of test depending on the device block. For instance, a memory block is tested differently to an I/O block. In this section, I only focus on the logic of the MPGA.

To test the logic of the Zelix MPGA, we need to generate a test vector and supply additional circuitry to make the test. In our case, the test vector generation can be realized with an automatic test pattern generation (ATPG). To allow all parts of the MPGA to be tested, we can use a scan-path design [18]. In a scan test, the flip-flops are implemented as a shift register chain, i.e. we can

Figure 3.15: Delay of the Zelix 4-in LUT with different load capacitances

imagine the scan path as a serial concatenation of flip-flops; the serial output of the scan path is analyzed to check for faults. The scan-path implementation needs a multiplexer to be introduced at the input of each flip-flop, so the final physical area is increased. In the Zelix MPGA, the introduction of a scan flip-flop represents an increment of 1.4% of the total tile area. The routing of the scan-path can be performed with the common routing or by dedicating a special routing; in the latter case the final area of the tile has to be increased by roughly 2%.

3.4 r-MPGA vs. Zelix MPGA

With 30 tracks per channel, the Zelix MPGA is 97% smaller in area than the r-MPGA; this reduction is valid only for a logic tile.

As the r-MPGA has no design tools to map a circuit, I can only compare delays and power consumptions from a conceptual point of view. The area difference

	r-MPGA	Zelix MPGA	Reduction
Delay	32ps	2ps	94%
Effective Capacitance	198fF	48fF	76%

Table 3.4: One-length wire delay, and capacitance

also implies a reduction in interconnect delay, since wire lengths are reduced by 94%. The delay for a 1-length line is about 32ps in the r-MPGA, and 2ps in the Zelix MPGA, see Table 3.4. Sometimes wires extend over more than one tile or can have big fan-out, so when interconnect delay becomes important a buffer needs to be inserted. For example, a wire that extends over six tiles has an associated delay (without buffer insertion) of 522ps in the r-MPGA, and 30ps in the Zelix MPGA. To compensate for this interconnect delay, the r-MPGA needs to have more buffers in its wiring than the Zelix MPGA.

The power consumption of a tile in an MPGA can be concentrated in the CLB or in the interconnect. The r-MPGA uses latches to configure the CLB. The static power consumption of a latch is 1 nW. We have with 208 latches per tile a static power consumption of $0.2\mu\text{W}/\text{tile}$. With 264 tiles it is $54.9\mu\text{W}$. Therefore, the static power consumption in the r-MPGA is in the order of μW . As dynamic power consumption is in the order of mW, the static power consumption due to the latches in the r-MPGA is not representative. The dynamic power consumption of the logic tile on the r-MPGA has to be similar to the Zelix MPGA. The routing information for the r-MPGA is the same as for the Virtex-II Pro, but it requires more wire-length than the Zelix MPGA because of the wire segmentation of the routing architecture. The capacitance is proportional to the wire length, and it is well known that the dynamic power consumption is proportional to the capacitance. Therefore, we can expect a significant reduction in the dynamic power consumption in the Zelix MPGA.

Because of the above results, I have chosen the Zelix MPGA as a target architecture for the conversion flow.

3.5 Zelix MPGA vs. Virtex-II Pro FPGA: Area

Using the values of Chapter 2, we can compare the final areas of the Zelix MPGA, and Virtex-II Pro. The shrink factor in the conversion means that the perimeter of the die core is reduced, so the I/O pads cannot be the same as in the original Virtex-II Pro. Mask programming also implies an area reduction in the memory and multiplier blocks; I have assumed a 10% area reduction, as in [70]. For example, the XC2VP2 has 204 I/O pads (see Table 3.5), so it is only possible to have 70 I/O pads in the Zelix MPGA. The total area reduction is 57.55% with the XC2VP2 device, and 63.50% with the XC2VP125 device. If we use 40 instead of 30 tracks per channel in the Zelix MPGA, the area reduction is 54.67% in the XC2VP2 device, and 60.34% in the XCVP125 device.

In this comparison only one ring of I/O pads is assumed. Therefore, the number of I/O pads in the Zelix MPGA is smaller than in the FPGA. But the number of I/Os can be important for a direct replacement of the FPGA if we want to maintain the same PCB board. If we want to have the same number of I/O pads in the Zelix MPGA, two I/O rings are necessary. For example, using all the 204 pads of the XC2VP2 with 30 tracks per channel, the area reduction is 51.5%.

	#	Zelix-MPGA		Virtex-II Pro		Reduction (%)
		Area mm ²	(%)	Area mm ²	(%)	
Logic	264	1.89	28.65	10.56	67.86	82.08
Memory	12	1.94	29.43	2.16	13.88	10
Mult.	12	0.65	9.81	0.72	4.63	10
PLL	4	1.17	17.66	1.17	7.50	0
I/O pads	70 (204)	0.95	14.45	0.95	6.13	0
Total		6.60	100	15.56	100	57.55

Table 3.5: Area comparison with XC2VP2 architecture (30 tracks per channel).

	#	Zelix-MPGA		Virtex-II Pro		Reduction (%)
		Area mm ²	(%)	Area mm ²	(%)	
Logic	13900	74.73	36.56	417	74.46	82.08
Memory	556	90.07	44.07	100.08	17.87	10
Mult.	556	30.02	14.69	33.36	5.96	10
PLL	12	3.50	1.71	3.50	0.62	0
I/O pads	446 (1200)	6.08	2.97	6.08	1.09	0
Total		204.40	100	560.02	100	63.50

Table 3.6: Area comparison with XC2VP125 architecture (30 tracks per channel).

3.6 Discussion: Shift Register, and SRAM in the Zelix-LUT

One limitation of the implemented LUT on the Zelix MPGA is the shift register. The configuration of the Zelix-LUT does not have registers, because is mask-programmed. In the Virtex-II Pro FPGA, the registers of the LUT can be used for implementing shift registers or SRAM memory. A rough estimation of the area overhead of adding registers is provided in this section.

The schematic for the shift register is shown in Figure 3.16. The memory cell is shown in Figure 3.17. Using the latches of the Infineon 130nm standard cells, the resulting tile area is $8038\mu\text{m}^2$. This represents an area reduction of 73.09% compared with the Virtex-II Pro.

Figure 3.16: LUT with shift register

Figure 3.17: Memory cell for shift register

Re-use of the shift registers for implementing SRAM functionality needs an added decoder. Figure 3.18 shows the schematic of the LUT with shift and SRAM functionality. Figure 3.19 shows the memory cell necessary for this solution. The final tile area is estimated at $9587\mu\text{m}^2$, representing an area reduction of 68.04% compared with the Virtex-II Pro.

Finally, I have compared the device areas, as in the previous section. Ta-

Figure 3.18: LUT with shift register, and SRAM memory functionality

Figure 3.19: Memory cell for shift register, and SRAM memory

	#	Zelix-MPGA		Virtex-II Pro		Reduction (%)
		Area mm ²	(%)	Area mm ²	(%)	
Logic	264	3.37	41.73	10.56	67.86	68.04
Memory	12	1.94	24.04	2.16	13.88	10
Mult.	12	0.65	8.01	0.72	4.63	10
PLL	4	1.17	14.42	1.17	7.50	0
I/O pads	70 (204)	0.95	11.80	0.95	6.13	0
Total		8.09	100	15.56	100	48.02

Table 3.7: Area comparison with XC2VP2 architecture (with shift register, and SRAM in the Zelix-LUT; 30 tracks per channel).

ble 3.7 and Table 3.8 show the results for the smallest and largest devices. The area reduction is around 50%.

3.7 Summary: Area

In this chapter, the r-MPGA, and the Zelix MPGA were described. The r-MPGA is a direct migration from the Virtex-II Pro FPGA. The idea behind the r-MPGA is to replace the programmable latches of the Virtex-II Pro FPGA by metallization. Hence the hierarchical routing architecture of the Virtex-II

	#	Zelix-MPGA		Virtex-II Pro		Reduction (%)
		Area mm ²	(%)	Area mm ²	(%)	
Logic	13900	133.26	50.68	417	74.46	68.04
Memory	556	90.07	34.26	100.08	17.87	10
Mult.	556	30.02	11.42	33.36	5.96	10
PLL	12	3.50	1.33	3.50	0.62	0
I/O pads	446 (1200)	6.08	2.31	6.08	1.09	0
Total		262.93	100	560.02	100	53.05

Table 3.8: Area comparison with XC2VP125 architecture (with shift register, and SRAM in the Zelix-LUT; 30 tracks per channel).

Pro FPGA is maintained. To reduce the high penalty area of the r-MPGA, the Zelix MPGA was created. The Zelix MPGA has a new regular interconnect architecture. Compared with the Virtex-II Pro FPGA, the Zelix MPGA has 82% area reduction for the programmable logic, and up- to 64% area reduction for the whole device.

Chapter 4

Zelix MPGA Conversion Design Flow

4.1 Introduction

This chapter describes the design flow for converting a Xilinx Virtex-II Pro FPGA prototype design into one using the Zelix MPGA device. It is important when developing a new MPGA architecture to ensure compatibility with the flow tools; architectural features that tools cannot handle are not useful. The tools, and the architecture must be developed together. The best strategy is to choose or program the tools as flexible as possible. The inherent scalability of the MPGA devices helps in the tool-architecture design process.

The goal of the conversion flow is to have the same (or better) timing performance for the MPGA compared with an FPGA. The previous chapter has shown that the Zelix MPGA logic is the same as the Xilinx Virtex-II Pro FPGA, so the main difference is in the delay due to the interconnect. To verify that we reach the timing goal, I have used static timing analysis (STA). In STA, the timing analysis is carried out in an input-independent manner, and looks for the worst-case delay of the circuit over all possible input combinations [69]. STA is used commonly as a part of the sign-off methodology for a digital silicon device. Our conversion flow checks the delay of all the

data-paths with STA; the same check using circuit-level simulations would be rather complex and time expensive, and is therefore prohibitive.

For the Virtex-II Pro I use the Xilinx ISE design environment. After our designed circuit is placed and routed, the Xilinx ISE creates two files for the conversion flow. One is a verilog gate-level netlist (in terms of LUTs, AND-gates, and so on), and the other is a Xilinx Description Language (XDL) file. The XDL file contains the annotated circuit placements.

This chapter is organized as follows: Section 4.2 gives an introduction to the timing problems of a conversion flow. Section 4.3 presents the MPGA design flow. In Section 4.4 the timing problems, and solutions associated with data arrival times are discussed. Section 4.5 discusses the addition of metallization to the Zelix MPGA. Section 4.6 summarizes the chapter.

4.2 Timing Problems in the Conversion Flow

The conversion to an MPGA has two timing problems: data arrival time slack, and hold time violation.

Data Arrival Time Slack

As depicted in Figure 4.1, the data arrival time (DAT) is the delay of a combinational logic path. The DAT end points are the data input of a flip-flop or the output pin. The DAT slack¹ is the data arrival delay difference between the FPGA, and the MPGA,

$$slack_{DAT} = FPGA_{DAT} - MPGA_{DAT} \quad (4.1)$$

A negative slack means that the data arrival time in the FPGA is faster than that of the MPGA. As the MPGA is smaller than the FPGA, we can incorrectly think that the interconnect, and therefore the delay, is shorter. The routing of the MPGA is different from that in the FPGA. A net routed in the MPGA

¹I assume that the FPGA DAT slack, i.e. "DAT - specified critical path", is always positive or equal zero.

Figure 4.1: Data arrival time points. Adapted from [79], page 1.11

can be longer, and have more delay than in an FPGA.

In practical terms, the DAT is only defined for the critical path, so the critical path delay of the FPGA may be larger than the critical path delay of the MPGA. In Section 4.4 I will discuss the data arrival time problems in the Zelix MPGA.

DAT Time, and Set-up Violation

A set-up violation occurs when the clock of the flip-flop rises before the data arrives at the input of the flip-flop. This happens when the setup time slack $slack_{SetupTime}$ is negative:

$$slack_{SetupTime} = T_{clk} + clock\ delay - (logic\ path\ delay + FF1_{delay} + FF2_{setup}) \quad (4.2)$$

T_{clk} is the clock period, $logic\ path\ delay$ is the delay from the output of the first flip-flop to the second flip-flop, $FF1_{delay}$ is the delay of the first flip-flop, and $FF2_{setup}$ is the setup time of the second flip-flop. Set-up violations can be controlled by the clock period, i.e. there are no set-up violations with a long clock period. Note that the clock frequency is limited by the critical path. In usual designs, the clock frequency is fixed. In the Zelix conversion flow, the frequency is fixed by the FPGA, therefore, in the conversion flow, the problem of solving a negative $slack_{DAT}$ is exactly the same as solving the set-up time

Figure 4.2: Hold time analysis. Adapted from [4], page 22-11

violations.

Hold Time Violation

A hold time violation happens when the clock timing propagation is slower than the data path delay. We can define the hold time slack as,

$$slack_{HoldTime} = data\ path\ delay - clock\ delay - FF2_{HoldTime} \quad (4.3)$$

$FF2_{HoldTime}$ is the hold time of the second flip-flop, and clock delay is the time difference between the clock arriving time of the first and second flip-flops, (see Figure 4.2). The data path delay includes the delays of the launch flip-flop. There is a hold time violation when $slack_{HoldTime}$ is negative. We can re-write the equation 4.3 as,

$$slack_{HoldTime} = logic\ path\ delay + FF1_{delay} - clock\ delay - FF2_{HoldTime} \quad (4.4)$$

$FF1_{delay}$ is the delay of the first flip-flop, and $logic\ path\ delay$ is the delay from the output of the first flip-flop to the second flip-flop. The smallest delay of a typical flip-flop used in the Zelix MPGA is around 100ps. The flip-flop hold time is around 25ps. We see that the delay caused by the first flip-flop is going to be compensated by the hold time of the second flip-flop. The maximum clock delay is the clock skew; the clock skew of the Zelix MPGA is 130ps; delays of a simple gate without interconnect delay (e.g. AND gate) are

around 130ps. Therefore, we may look at the case when the data path is pure interconnect (i.e. without any logic between the two flip-flops). Flip-flop to flip-flop interconnect can lie between the global interconnect and the intra-tile interconnect. The global interconnect delay between two flip-flops is always going to be larger than the clock delay, because the global interconnect has a staggered pattern, and the clock has a H- network. I conclude that hold time violations can only happen with the intra-tile interconnect. In this case the clock delay can be the same as the path delay. But no hold-time violations are expected because the delay of the first flip-flop is going to compensate for the final slack. We note that the calculation of the hold time violations in the MPGA is independent of the delays of the FPGA. It is possible to solve a hold time violation by inserting a buffer in the path delay.

Timing Considerations

I have assumed in this thesis that the designed circuit is well constrained, i.e. the input (or output) delays from the input pin (or output register) to the input register (or output pin) is well defined. Also, I assume that the conversion is being made from a synchronous circuit with only one clock domain. The mapping of asynchronous circuits or multi-clock domains can produce hold time violations.

Methodology to Solve the Timing Problems

A methodology to solve the timing problems can only be established through experience and experimental work; it consists of two parts: timing analysis, and delay reduction techniques. Both parts may be integrated in the conversion design flow. The conversion design flow proposed in this thesis has three basic steps:

1. **Mapping the Circuit:** Map the FPGA gate netlist without buffers into the Zelix MPGA. Insert buffers optimally into all the nets to improve timing.

2. **Verification:** Verify the timing integrity and the hold time.
3. **Layout Creation:** Create the programmable masks of the layout.

In the first step, timing analysis is done and buffer insertion is used to improve the timing of the converted design. In the second step, the delays are analyzed. The final step is the creation of the tape-out. It seems a priori that the timing integrity cannot always be satisfied with this conversion flow. I will show in this chapter, and in the experimental work (Chapter 5) what to do in order to guarantee timing integrity.

4.3 Zelix MPGA Design Flow

The design flow of the MPGA architecture is shown in Figure 4.3. This design flow consists of: manipulations or modifications of the original FPGA design netlist, gate placement, routing, verification, and tape-out creation. The following sections will explain these phases in detail.

4.3.1 Netlist Modifications

I have re-used the data from the FPGA flow as much as possible to simplify the conversion. The Zelix MPGA flow starts from the Xilinx STA gate-level netlist, whose original purpose is timing analysis. After a detailed study, I observed that this netlist could be used for the physical implementation. This netlist models gates for configurable signal routing by means of buffers that are used to annotate programmable interconnect delays. A Perl-script removes all buffers (including the hardwired buffers); in a later step of the flow, the necessary buffers for the interconnect are added. The elements of the resulting netlist are shown in Table 4.1. The second column shows the resource number in a tile. The inverters are used for inverting the clock and control signals of FFs.

Figure 4.3: Zelix MPGA design flow

4.3.2 Zelix MPGA Placement

In an FPGA, the netlist gates are usually packed into logic blocks or slices. The placement step assigns the packed logic to specific logic slice locations in the prefabricated array. A simulated annealing algorithm is used by FPGAs to place the circuits [75]. The Xilinx ISE environment for the Virtex-II Pro uses a timing-driven algorithm [59]. It is claimed that the Virtex-II Pro synthesis tool iterates between technology mapping and placement to improve the timing [49].

Two different types of placement can be found in MPGAs: fine-grain and block-level placement. The fine-grain placement is at the gate level, and uses a placement tool similar to standard-cell semi-custom tools. The block-level placement is similar to FPGA placement, while the block-level placement needs a packing step (to group the gates into slices), which is not needed in the fine-grain placement; hence the highest area density is obtained with

Cell	Instances in a Tile
4-input LUT	8
FFs	8
2:1 Mux	16
AND2	8
OR2	4
XOR	8
INV	10

Table 4.1: MPGA netlist elements

fine-grain placement. To my best knowledge there has been no published work that investigates how best the placement can be provided. The goal for MPGA placement is to create a placed configuration of logic gates that can be successfully interconnected in a subsequent routing step. So the best solution for the placement depends on the routing architecture and the placement algorithm. In [28] DeHon demonstrates empirically that, in pre-defined routing architectures, higher logic density is less area efficient, because higher density implies more routing resources, i.e. more tracks per channel. Also, [85] shows that by de-populating tiles, thus lowering density, it is possible to obtain better timing performance. In the Zelix MPGA, the routing architecture is more flexible than in an FPGA (e.g. the switch matrix has 100% routability). This means that an MPGA can have higher density placement than an FPGA. In non pre-defined routing architectures, such as CMOS- based MPGAs, and Cell-based MPGAs, 100% density placement is also prohibitive (because the routing has to be satisfied with a limited number of metal layers). I conclude that although MPGA placement can be denser than FPGA placement, this packing and placement approach will be adequate for the Zelix MPGA. Further studies on the Zelix MPGA placement will have to be done to validate this statement.

For my project I have imported the original placement used in the Xilinx ISE flow. It would also be possible to do the MPGA placement with another placement tool, such as Cadence Encounter [20]. One advantage of using the Xilinx placement is that we can compare our MPGA routing with the Virtex-II Pro routing. The Xilinx placement information is extracted from the Xilinx Description Language (XDL) file. To match the names of the Xilinx gate-level netlist and the XDL file, a deep analysis of the Virtex-II Pro architecture has been done. The placement information obtained is at tile level, not at slice level. The placement information on a slice is more difficult to obtain, and is error prone. By importing tile level placement, instead of slice level, in the MPGA a difference for the local interconnect delay may exist. In the current Zelix implementation, the CLB has a dimension of $39.2\mu\text{m} \times 64\mu\text{m}$. A wire of length $64\mu\text{m}$ has a capacitance of around 13fF (in the order of an input pin, i.e. 5-10fF), so the delay difference due to the capacity introduced with the logic placement at the top or bottom of the CLB is low compared with the total net delay.

4.3.3 Zelix MPGA Routing

Routing is the process of identifying which routing segments should be used to create connected paths from net sources to net sinks for all nets in a design. In this section I will explain the data structures requirement for the Zelix MPGA routing and then I will describe the MPGA routing algorithm.

Routing-resource Graph

The routing algorithm is independent of the MPGA architecture; the architecture can be defined without worrying about the CAD algorithms. In the source code, the router and the architecture are two independent files that are linked at compile time. The routing algorithm searches a directed graph, the

Figure 4.4: Zelix MPGA routing-resource graph

routing-resource graph, for a solution. This directed graph is a representation of the routing architecture. Specification of all the connections of the architecture needed for routing is practically impossible; a detailed routing description in C-language can be of more than 10000 lines. An alternative is to create a high-level description of the MPGA, and then generate the routing-resource graph with a C or Perl program [14, 17, 24, 51]. The high level description is based on fundamental architectural parameters, such as number of tracks per channel, number of LUTs in a CLB, type of switch matrix used, etc.

The approach used in this thesis is a medium-level description; neither a reduced set of parameters nor the complete graph. I have written a basic architectural definition for a tile and an I/O pad. The program replicates this tile and pad architectural definition into the routing graph. The tiles arranged around the array have special interconnect features; the original tile needs modifications. The parameters that the program uses are shown in Table 4.2. With a high maximum number, the memory resource increases, and therefore so does the compile time for the sources. The nodes of the routing-resource graphs are the wires or pins of the Zelix MPGA. There are four different types of nodes: outpin, inpin, xwire, and ywire. The outpin and inpin are the output

Parameter	Default value
Number of columns	22
Number of rows	16
Number of vertical tracks per channel	30
Number of horizontal tracks per channel	30
Number of nodes in the routing graph	88774
Maximum number of inpins in a CLB	150
Maximum number of outpins in a CLB	90
Maximum number of inpins in an I/O pad	20
Maximum number of outpins in an I/O pad	20
Maximum number of nets	2090

Table 4.2: Routing-resource graph parameters

and input pins of a CLB respectively. The xwire and ywire are the horizontal and vertical tracks of a channel respectively. The edges of the graph are switches from one routing resource to another; each edge can have several real switches. Figure 4.4 shows part of the routing-resource graph belonging to a tile. The graph starts from an outpin. In the example, the outpin has five edges, i.e., the connections to north, south, east, and west wires for global routing, and a connection with an inpin of the same tile (local routing). A wire resource connects two tiles. As mentioned in Chapter 3, the input/output crossbar, and the switch matrix have 100% population of possible via connections. This feature simplifies the routing graph, and is good for saving memory. Two wires in the same channel are equivalent; the routing tool makes no distinction which exact track in the same channel is used. In FPGAs each track in a channel is different. To save area, FPGAs have different track segmentations, and the switch matrix, and the input/output crossbar are not 100% populated with switches. I have assigned a capacity to each node, which is the maximum

number of nets connected to that node. Inpin and outpin nodes have capacity of one, since only one net may be used. Wire nodes have larger capacities; in reality, the maximum number of tracks per channel. Therefore, all nodes that represent vertical tracks have the same capacity. The same holds for the horizontal wire nodes.

Zelix MPGA Routing Algorithm

The developed Zelix MPGA routing algorithm is based on the Lee maze-routing algorithm [44]; it consists of three steps: wave expansion, traceback, and clean-up. Figure 4.5 shows the algorithm graphically. The wave expansion uses the depth breadth search [22]. The algorithm starts from the source of the net. It finds all the paths reachable at distance one, using depth breadth search, i.e. all the paths that are just one unit in total length. After that, all the paths are labeled with the current distance. The previous two steps are repeated until the sink is reached. The traceback phase follows the labeled routing resource from the sink to the source selecting a shortest path (any shortest path). The routing resources used are marked. Finally a clean-up for the grid is done. A maze router will always find the shortest path between the source, and the target if such path exists. It may be implemented in $O(n^2)$ time where n is the number of routing resources [52].

The Lee maze-route algorithm is only for two-point nets. For multi-point nets, the strategy is: use the Lee maze route algorithm to find a path from source to nearest target. Then, expand simultaneously from all of these resources to find the shortest path from the existing route to the next target.

Dijkstra's algorithm is used to add a cost function [29]. In this variant of Lee's algorithm, each resource (segment/pin) has a cost (or weight) associated. In the wave expansion the cost is added.

Another feature introduced in our algorithm is to restrict the wave expansion to a boundary box, to speed up the processing time of the algorithm.

Figure 4.5: Maze router description

(a) Adding a Sink

(b) Multi Sink routing

(c) Routing with cost (Dijkstra Algorithm)

(d) Boundary Box for Wave Expansion

(e) Pathfinder: Congestion Problem

(f) Pathfinder: Congestion Solved

Figure 4.6: Extensions of the Lee Maze Router

The Lee maze routing depends strongly on the order of the routed nets. The Pathfinder routing offers a solution for finding an optimal order [30]. In the Pathfinder algorithm, during each routing iteration, the nets are routed using the maze-routing algorithm with cost function. The Pathfinder algorithm allows overuse (congestion) of nodes (with either a wire or a logic block input pin). If congestion exists (illegal routing), the cost of the congested resource is updated, based on the amount of overuse. One iteration of the router consists of sequentially ripping-up and re-routing (by the lowest cost found) every net in the circuit. Algorithm 4.1 shows the pseudo code of the Pathfinder algorithm. RT_i is the routing tree, S_i is one net, n_s represent the node s , and P_j is the tree with shortest path between RT_i and n_s . The Pathfinder algorithm ends when a solution is found without routing resource congestions, or when the maximum number of iterations is reached. The shortest path algorithm is described in Algorithm 4.2. Q is the current tree solution, n is the index of the node, and m is a variable index of a node. The shortest path algorithm is the Lee maze- router with a cost function for one net.

The cost function of the shortest path algorithm using routing resource n is,

Algorithm 4.1 Pathfinder Algorithm

```

1.  while (there are shared routing resources){
2.      for (all the nets,  $S_i$ ){
3.          if ( $S_i$  has shared nodes){
4.              rip up  $S_i$ 
5.               $RT_i \leftarrow n_s$  (source node of  $S_i$ )
6.              while (there are sinks of  $S_i$  unconnected to  $n_s$ ){
7.                   $P_j \leftarrow \text{find-shortest-path}$  ( $RT_i, n_s$ ) (see Algorithm 4.2)
8.                  add  $P_j$  to  $RT_i$ 
9.              }
10.         update first, and second-order costs of  $RT_i$  nodes
11.     }
12. }

```

Algorithm 4.2 Find Shortest Path

```

1.  foreach (node  $n$  in  $RT$ ){
2.      enqueue  $n$  onto  $Q$  with key 0
3.  }
4.  while (a new sink of  $n_s$  has not been found{
5.      dequeue node,  $m$ , with lowest key from  $Q$ 
6.      if ( $m$  was not previously dequeued){
7.          foreach (neighbour  $n$  of  $m$ ){
8.              enqueue  $n$  on  $Q$  with cost of  $n$ +key of  $m$ 
9.          }
10.     }
11. }
```

Routing Resource	Default value
Output Pin	5.84388e-10
Input Pin	5.55169e-10
Channel X	5.84388e-10
Channel Y	5.84388e-10

Table 4.3: MPGA base cost (b_n) values.

$$c_n = b_n \times p_n \times h_n, \quad (4.5)$$

where b_n is the base cost, p_n is the present cost, and h_n is the accumulated cost. The base cost b_n depends on the routing resource type, and is used for penalizing bends. The current values of b_n are shown in table 4.3. The present cost p_n is defined to be a function of the difference between the current occupancy of the node: $occupancy(n)$, and the capacity of the node: $capacity(n)$ as shown in equation 4.6. As p_n is always equal or greater than 1, a maximum function (MAX) with zero is used. The routing schedule of the routing p_{fac} has different values depending on the iterations (see table 4.4). The p_{fac} term

Parameter	Default value
p_{fac} (first iteration)	0
p_{fac} (second iteration)	0.5
p_{fac} (other iterations)	1.5
h_{fac}	1.5

Table 4.4: Cost function parameters

is 0 for the first routing iterations, and is gradually increased in successive iterations. Hence, in the first iteration the nets are routed by the shortest path found. In the successive iterations the routing tries to avoid the congestion.

$$p_n = 1 + MAX(0, [occupancy(n) - capacity(n)] \times p_{fac}), \quad (4.6)$$

The accumulation cost h_n has the past congestion values; h_n is defined by equation 4.7:

$$h(n, i) = \begin{cases} 1 & i = 1 \\ h(n, i - 1) + MAX(0, [occupancy(n) - capacity(n)] \times h_{fac}) & i > 1 \end{cases} \quad (4.7)$$

where i is the iteration index. The h_{fac} is a factor to ponderate the accumulated cost. h_{fac} is a constant parameter between 0.2, and 2. Its value is shown in Table 4.4.

The router tool has been programmed in C using software fragments from the VPR FPGA tool [15]. My variant of the Pathfinder maze router algorithm implements a routability-driven router instead of a timing-driven router. We can have a timing-driven router adding the Elmore delay metric into the cost function of the router. I think that no major benefits can be derived with this timing-driven router. The majority of the nets (60% ~ 70%) have only one sink [21]. For two-pin nets the routing solution is not different using the routability-driven router. Remember that the routability-driven router looks

for the shortest path between the sources and the sinks. The shortest path in a two-pin net is the path with shortest delay. For multiple sink nets the shortest path does not always mean the shortest delay; but in most cases the short paths lead to almost the shortest delay. The timing-driven router is practically only necessary when we mix buffered with unbuffered wires [15]; however in our conversion flow, the buffer insertion is done at a later stage. The complexity of a timing-driven router is higher than of a routability-driven one. Also timing-driven routers need a higher number of tracks per channel than routability-driven routers. Because the regular interconnect architecture has a penalty in terms of area, we want to optimize the number of wires for the critical path. Notice that the conversion flow must guarantee the same timing as in an FPGA, not a better performance. For these reasons I did not add the timing optimization to our MPGA router. In FPGAs, whether the interconnect-segmentation is pre-buffered or not, the timing-driven router is a good choice. FPGAs have not got the flexibility of inserting buffers as do the MPGAs.

It is possible to set the number of tracks per channel in the MPGA router. The number of tracks per vertical or horizontal channel can be different (in the VPR FPGA tool vertical and horizontal channels are the same). This feature is important because the MPGA layout of a tile is usually rectangular. So in one direction, there can be more tracks per channel.

4.3.4 Place and Route Visualization

A tool has been programmed to visualize the placement and routing of the MPGA. The EasyGL package has been used [12]. Figure 4.7 shows the placement, and Figure 4.8 the routing of one circuit. In the visualization, only the channel paths of the routed net are shown.

Figure 4.7: MPGA viewer: placement

4.3.5 Buffer Insertion

Interconnect delays become important in deep-submicron technologies. The delay of a wire is proportional to its intrinsic capacitance, and resistance. Moreover the wire delay depends quadratically on the wire length. If we reduce the capacitance or the resistance of a net, we reduce the delay. Buffer insertion is another common technique to reduce this delay: a buffer breaks the net into two smaller nets, and due to the length dependence being a power of two, the delay is reduced. The buffer isolates two nets, but itself adds a small delay to the total path. Buffer insertion in digital ICs can be divided into:

1. **Fanout optimization during logic synthesis.** The focus of this buffer insertion is delay optimization. A fanout model is used, since no routing information exists in logic synthesis.

Figure 4.8: MPGA viewer: routing

2. **Timing-driven buffer insertion after routing.** Here the target is timing performance, and also reducing the noise due to the interconnect. The buffers are inserted in critical nets where the specified time is violated. When a buffer is inserted because there is a significant resistance effect of a wire², the buffer insertion technique is called repeater insertion.
3. **Clock-tree buffering.** In this category, buffer insertion is for delay and skew minimization.

In Chapter 3, the use of buffers from the clock-tree for the Zelix MPGA has been explained. FPGAs have a pre-fabricated routing architecture with buffers. So, it is not usual for the synthesis tool to insert buffers. A buffer

²as a "golden rule", a wire resistance is significant when it is equal or greater than the resistance of the gate-driver [11]

function in an FPGA is implemented with a LUT. In the Zelix MPGA, I removed all the buffers in the Virtex- II Pro synthesis netlist. I had to perform buffer insertion after routing to compensate for long net delays caused by long wires or large fanout. I applied the dynamic programming van Ginneken's algorithm [87] for buffer insertion; this uses the Elmore model [31] for delay analysis. The algorithm finds the buffer positions in a RC-tree in such a way that the Elmore delay is minimal. The topology of the RC-tree is a Steiner tree. The complexity of the van Ginneken's algorithm is $O(n^2)$, where n is the number of possible buffer positions. Note that the van Ginneken's algorithm optimizes delay in the fanout and the long wires. The next two sections describe the Elmore delay and the van Ginneken's algorithm, in order to explain the buffer insertion in the Zelix MPGA.

The Elmore Delay Metric

The Elmore delay provides a useful technique for estimating the delay in interconnects. Let $h(t)$ be the impulse response of a node of a RC circuit. The transfer function $H(s)$ of $h(t)$ can be represented as:

$$H(s) = \int_0^\infty h(t) \cdot e^{-st} dt = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!} s^k \int_0^\infty t^k h(t) dt \quad (4.8)$$

The Taylor expansion of $H(s)$ at $s = 0$ is:

$$H(s) = m_0 + m_1 s + m_2 s^2 + \dots \quad (4.9)$$

where m_k are the moments. With the equations 4.8 and 4.9 the moments are given by:

$$m_k = \frac{(-1)^k}{k!} \int_0^\infty t^k h(t) dt, \quad (4.10)$$

for $k = 1, 2, \dots$

In [31] Elmore has shown that the mean of the impulse response $h(t)$ is the negative of the first moment ($k = 1$),

$$m_1 = \int_0^\infty t \cdot h(t) dt \quad (4.11)$$

The mean m_1 approximates the 50% delay of the step response. Let an RC interconnect be a rooted tree where each branch corresponds to a distributed capacitance to ground and a distributed resistance between nonground nodes. The Elmore delay from a source, s_0 to a sink, $sink_i$ in an RC interconnect is [66]:

$$t(s_0, sink_i) = \sum_{j \in Path(s_0, sink_i)}^n R_j \cdot \sum_{k \in downstream(j)} C_k, \quad (4.12)$$

where $Path(s_0, sink_i)$ is the unique path from source s_0 to sink $sink_i$, R_j is the resistance at node j , and the downstream capacitance at node j is the sum of all capacitances at any node k , such that the unique path in the tree from j to the root must pass through k . As example consider the RC interconnect shown in Figure 4.9. The Elmore delay in $sink_2$ is,

$$\begin{aligned} t(s_0, sink_2) &= R_1 \cdot (C_2 + C_3 + C_4 + C_5 + C_6 + C_7) \\ &\quad + R_3 \cdot (C_3 + C_4 + C_5 + C_6 + C_7) \\ &\quad + R_5 \cdot (C_5 + C_6 + C_7) + R_7 \cdot C_7 \end{aligned} \quad (4.13)$$

The Elmore delay of a single edge $e = (node_j, node_k)$ of the RC tree is,

$$t(e) = R(e) \left(\frac{C(e)}{2} + \sum_{k \in downstream(j), k \neq j} C_k \right), \quad (4.14)$$

where $R(e)$ and $C(e)$ ³ are the associated lumped resistance and capacitance of the edge e . The capacitance $C(e)$ is divided by two in the equation because I use a Pi model for the interconnect.

The Elmore delay metric is assumed to be accurate for performing interconnect CAD optimization, such as timing-driven routing or buffer insertion. But it is not accurate for circuit timing analysis, especially when there is a high degree of resistive shielding.

³The letter in the brackets stands for edge (e) or sink (s).

Figure 4.9: An example of an RC interconnect

The van Ginneken's Algorithm

The objective of the optimal buffer insertion is to find the buffer positions in the interconnect so that the timing requirements are met.

Definitions:

A net is a tree $T = (V, E)$, where $V = v_0 \cup V_s \cup V_n$, and $E \subset V \times V$. Vertex v_0 is the source, and the root of V , V_s is the set of sinks, and V_n is the set of internal tree nodes. Each sink has associated a capacitance $C(s)$, and a required arrival time $RAT(s)$. Each edge $e \in E$ has associated a capacitance $C(e)$ and resistance $R(e)$. Any vertex $v \in V$ has a $T(v)$ subtree downstream from v . Figure 4.10 shows an example of a net.

The delay of an inserted buffer b in a node v_i is:

$$D(v_i) = K(b) + R(b) \cdot \sum_{e \in \text{downstream}(v_i)} C_e, \quad (4.15)$$

where $K(b)$ is the intrinsic buffer delay, $R(b)$ is the buffer driver resistance.

Once we decide where to insert buffers in $T(v)$, we have a candidate, α , for

Figure 4.10: Tree representation of a net. S_0 is the source. $v_1, v_2, v_3, v_4, v_5, v_6$ are possible buffer positions. The downstream capacitance at v_2 is $C_1 + C_2 + C_3 + C_{12}$.

$T(v)$. The delay from v to sink $s \in T(v)$ is:

$$D(v, s, \alpha) = \sum_{e=(v_i, v_j)} (D(v_i) + D(e)) \quad (4.16)$$

If the node v_i has no buffer, then $D(v_i) = 0$.

The slack of a node v is defined as:

$$D(v, \alpha) = \min_{s \in T(v)} \left(RAT(s) - D(v, s, \alpha) \right) \quad (4.17)$$

Buffer Insertion Problem:

Given a routing tree $T = (V, E)$, sink capacitance $C(s)$, and $RAT(s)$ for each sink s , capacitance $C(e)$, and resistance $R(e)$ for each edge e , a possible buffer position f , and one buffer type, find a candidate α for T that maximizes the slack at the source.

Assumptions in the van Ginneken's Algorithm:

Each candidate, α , of a tree branch is represented by a (Q, C) pair, where Q is the slack, and C is the capacitance.

For two candidates α_1 , and α_2 of the same branch, if $Q(\alpha_1) < Q(\alpha_2)$, and

$C(\alpha_1) > C(\alpha_2)$, then α_1 is redundant.

For a routing tree with n buffer positions, there are at most $n+1$ non-redundant candidates.

Operations in the van Ginneken's Algorithm (see 4.3.5):

1. **Add a wire:** For each candidate α , subtract the wire delay from $Q(\alpha)$, and add a wire capacitance to $C(\alpha)$.
2. **Add a buffer:** At each possible buffer position, create a new candidate with a buffer.
3. **Merge Branches:** Merge candidates of two branches of the routing tree. If (Q, C) , and (Q', C') are the slack and the capacitance of two branches, then the slack, and the capacitance of the merged branches are $(\min(Q, Q'), C + C')$.

van Ginneken's Algorithm:

The van Ginneken's algorithm consists of the phases:

1. **Bottom-up phase:** During the first phase the function *bottom-up* computes all the possible buffer assignments for each node in the tree.
2. **Top-down back-tracing:** The second phase traces back the computations of the first phase that led to the buffer assignments with the best slack at the source.

The Bottom-up pseudo-code is shown in Algorithm 4.3. The algorithm starts in the root, and finishes when all the sinks are processed. When a sink is found, the function Bottom-up returns the slack, and the capacitance. If not, the function is called recursively to compute the options.

Algorithm 4.3 Bottom-up Function

-
1. **if** (found sink){
 2. $RETURN(Q, C_v)$
 3. }
 4. **else** {
 5. "compute options for subtrees"
 6. Bottom-Up(left(v))
 7. Bottom-Up(right(v))
 8. Join pairs of subtrees by merge operation
 9. Find best candidate among merged candidates to add a buffer
 10. Add parent wire to both types of candidates
 11. Prune inferior candidates from set of candidates
 12. Store candidates list for node v and return
 13. }
-

(a) Slack, and Capacities in Sources, and Sinks

(b) Add a Wire

(c) Add a Buffer

(d) Merge Branches

Figure 4.11: Van Ginneken's operations

Zelix MPGA's Buffer Insertion Algorithm

Our programmed algorithm is a generalized extension of the published van Ginneken's algorithm. For instance, in van Ginneken's paper, the nodes of the Steiner tree can have at most two leaves, in our case there are no leaf limitations. Van Ginneken's algorithm assumes that a set of buffer insertion candidate locations are predetermined for the given topology. The later is a van Ginneken's algorithm quality problem for ASIC semi-custom methodologies, where the buffer positions are unknown a priori [2]. In an MPGA the buffer positions are predefined, but must be planned [105]. The Zelix MPGA architecture allows the use of buffers at the end points of the interconnect one-segment wire.

4.3.6 Verification

Interconnect Model

After place, route and buffer insertion, we can import our design to a physical layout. With the physical layout it is possible to perform extraction (e.g. with Cadence ASSURA tool [19]), and then run circuit-level simulations (e.g. with Synopsys NanoSim [80]). The problem with this approach is the associated long extraction and simulation time. To reduce the length of simulations, the solution is to use a STA analysis without RC extraction. In this level of abstraction, the load model and the net delay are separated. In addition, wave shapes are abstracted as saturated ramps, characterized by a delay to the 50% point and a slew extrapolated from the real 10%-90% or 20%-80% slew [73]. To perform the STA analysis without RC extraction an interconnect model is needed. The interconnect model is specified with the DSPF format [76]. The regularity of MPGA routing architecture simplifies our interconnect model. The interconnect parasitic model has five different elements: wire, input crossbar, output crossbar, cell input/output pins, input/output pads. All the cases

are modeled as worst case scenarios. For the input crossbar, the assumed worst case is when one wire comes to the switch matrix and goes to the input cell: the same parameters are assumed for the wire. Note that the MPGA has a one-segment global routing architecture. The length of the wire is the distance from one tile to its nearest neighbor. The global routing architecture needs only a model for two wires (vertical, and horizontal). A wire has three physical characteristics: capacitance, resistance, and inductance [26]. Inductive effects are significant in a high signal bandwidth regime (or for weak rise times) [39]. If the product of inductance, and angular frequency, which corresponds to $2\pi \cdot 0.35/T_{rise}$ is comparable to the line resistance, a first-order statement that inductive effects are important can be made [77]. In the MPGA a common rise time for 100 MHz or 200 MHz clock signal is around 120 ps. The typical maximum clock frequency in a 130nm Virtex-II Pro FPGA is around 200MHz. A one-length wire has a resistance of 70.8Ω , a net with 100 wires is going to be around 7 k Ω . So the inductance has no large effect in the Zelix MPGA. I expect a good accuracy with a model using wire resistance and wire capacitance. A one-length wire is modeled as a distributed π section. The values of the wire resistance, and capacitance are calculated using the Infineon 130nm CMOS six-layer (copper) process technology design manual [37]. The wire resistance and capacitance are calculated with the metal 3, and a width of 200 nm. In the total interconnect capacitance calculation, the fringing, coupling, and plate capacitances are included. A via resistance between metal 2, and metal 3 is about 0.9 Ω , and therefore neglected. For the input and output pins, Infineon's data-book is used. A Perl-script creates an interconnect RC tree with the routing information (a DSPF file). Figure 4.12 shows an example of a net with our interconnect model.

Figure 4.12: Example of the interconnect model

Static Timing Analysis

Once we have the interconnect information of the Zelix MPGA, we can proceed with the STA analysis. The STA tool used is Synopsys PrimeTime [78]. When PrimeTime reads the parasitic DSPF file, an internal interconnect model is created. The flow guarantees that all the data-path delays are reported with a PrimeTime TCL-script. Although the MPGA and FPGA gates (LUTs, FFs,...) are equal, the gate load capacity is different, because of the different interconnects, so the MPGA and FPGA gate delays are different. The delays are calculated from the output of one flip-flop or input pin to the input of a FF, or the output pin. Synopsys PrimeTime always reports the worst-case data-path of all the possible paths that have the input FF. Due to the reduced area of the MPGA array, I also expect shorter interconnect delays compared to an FPGA, in general. When an MPGA data-path has more delay than the corresponding FPGA data-path, there are two possible solutions: first to check if the mapped circuit still satisfies the specifications (i.e. less than the delay in critical path), and second to try to reduce the delay. One technique to

	Zelix MPGA Circuit Simul.	Zelix MPGA STA (with our model)
Net1	6.72ps	14ps
Net2	6.52ps(T2)/8.35ps(T3)	12ps(T2)/17ps(T3)
Net3	1.31ps	1.55ps

Table 4.5: Simulation of three nets

reduce the delay is buffer insertion; another technique is re-routing the circuit. Re-routing implies re-ordering nets, and introducing different penalties. I compare circuit level simulations, and STA to verify the interconnect model. Three cases are studied: a fan-out one net that expands two tiles (Net1), a fan-out two net where the two inpins expand one and two tiles (T2, T3) respectively (Net2), and a fan-out one net that is routed inside a tile (Net3). Table 4.5 shows the results. We can see that our simple interconnect model always has worse delay than the circuit level simulations. The latter is caused in part because the model always uses the full capacity and resistance of a one-length wire; the real capacity and resistance are a proportion of a one-length wire.

Discussion: Physical Verification

If we compare the Altera HardCopy flow (see Figure 2.8 in Chapter 2) with our conversion flow (see Figure 4.3), we can see a major difference: physical verification. In current digital IC designs, the verification requested for final sign-off is STA analysis. As previously discussed, STA analysis needs a library and the interconnect parasitic information. As the Altera HardCopy routes the design with a semi-custom tool, RC extraction has to be done from the layout. RC extraction takes a long time depending on the circuit size. For instance, the RC extraction from the layout took 10 days in one part of the Infineon

TriCore processor [42]. The interconnect parasitic information in the Zelix MPGA is obtained by means of a model. This means that an RC extraction is needed only once from the layout to create the parasitic file. After routing, a Perl-script creates the parasitics in less than one hour. We can see a time advantage in physical verification with the Zelix MPGA.

4.3.7 Layout Creation

The last step in the conversion flow is the creation of the programming layers of the Zelix MPGA layout. Perl-scripts have been programmed to perform the interface between the flow presented in the previous sections and the Cadence tools of the physical layout. Figure 4.13 shows an example of the mask programming of three tiles, in which the following elements are considered: the routed interconnections, LUT, and other configurations, clock network, and the power grid. The routed interconnection information is generated after the routing, and the buffer insertion of the conversion flow. LUT, and other configurations are annotated in the Xilinx verilog netlist. The configurations consist of the inputs of the related logic tied to V_{DD} or V_{SS} . In the case of the LUT, the channels between the input pins, and V_{DD} or V_{SS} are pre-planned inside the CLB. For the other configurations the I/O crossbar is used. The clock network uses the flip-flop positions of the mapped circuit. The power grid is automatically generated without additional information.

Once we have the physical layout, RC extractions can be performed and circuit-level simulations with annotated parasitics run are performed. A final step after a successful verification consists in the creation of the final tape-out to fabricate the device.

Figure 4.13: Three tiles layout.

4.4 Discussion: Data Arrival Time Slack

The slack of the data arrival time depends on the conversion flow. In practice, there is one parameter that is important: the number of tracks per channel. Two different cases can be evaluated:

Large number of tracks per channel

When we have a larger number of tracks per channel, the resulting routing is always a short path between the source and the sink. In this case, the MPGA paths are always shorter than in an FPGA. As the delay depends on the length of the path, the delay in the MPGA is always smaller than in an FPGA. Note that the nets in MPGA have to be buffered where needed, i.e. nets with big fan-out, and long wires. The problem of having a large number of tracks per channel is that the physical area is large. Figure 4.14 illustrates a

Figure 4.14: Short path net in Zelix MPGA

net routed with a short path in the Zelix MPGA (a short path in the Virtex-II Pro FPGA is similar).

Small number of tracks per channel

With a small number of tracks per channel the area is small, but there can be cases where a path in an FPGA is smaller than in an MPGA. Therefore, the MPGA can have a larger delay: Figure 4.15 shows this case. Unfortunately, if we compare delays net by net, a better timing in the MPGA cannot always be satisfied. It is an advantage to think how the timing in circuits is studied. The DAT time usually involves several nets, and logic paths. We can accept that one net of the MPGA DAT is larger than in the FPGA, but other nets are going to be smaller. But note that the important case to analyze is the worst DAT, i.e. the critical path.

Although I expect that in practically all real cases the critical path in the FPGA is going to be worse than in an MPGA, timing integrity cannot always

Figure 4.15: Long path net in Zelix MPGA

be achieved. If this happens we can still solve the timing integrity problem. First, we can analyze the critical path, and re-route the circuit with optimization in the nets of the critical path. Second, we can use more metal layers. Adding a metal layer has the same effect as increasing the number of tracks per channel. When a metal layer is added, the buffers available per wire decrease. As the utilization of buffers per wire is low, this is not a problem, but a special script has to be written. This case is discussed in the next section.

4.5 Discussion: Adding Metallization

The addition of metallization is worth considering when the number of tracks per channel to route our circuit is not enough. Also it can be useful to reduce the area for the MPGA master. The optimal case is the use of a minimum number of metal layers, i.e. if more metallization is needed, only one metal should be added. There is another problem associated with more metallization: buffer insertion. When we have more metallization it is difficult to access the

lower levels where the buffers are located. First, I study the case without buffer insertion, and assume (as is the case in the current Zelix implementation) that the top-level routing is performed with metal three and four, and we want to add one additional metal layer (i.e. metal five). The addition of one metal layer represents modifications on the switch matrix. As show in Figure 4.16, if we choose the correct order of metallization there is no conflict. The problem is that in some cases the order for one metal is fixed, and the upper metallization cannot gain access to the lower metallization (see Figure 4.17). In this case we can use more metallization, as shown in Figure 4.18.

The insertion of buffers requires a methodology. The steps of this methodology are:

1. Insert wires with buffers.
2. Insert wires with metal three, and metal four
3. Insert wires with metal five
4. If there is blockage:
 - (a) Try to move metal three.
 - (b) or use metal six.

In this methodology, the metallization is created by parts, depending on its criticality, instead of net by net. The pins of the buffers are positioned in metal three, i.e. in our scheme a buffer is always accessed with metal four. As the buffer insertion is the most constrained problem, we start inserting it; metals three and four are the easiest to use. So I use them as much as possible. When this is not possible, I use metal five. Metal five can be blocked by metal four, therefore we can try to avoid the blockage moving the metal three of the switch matrix to another position or if this is not possible we can always use metal six to avoid the blockage (but at more cost).

Figure 4.16: Adding metal five correctly

4.6 Summary

In this chapter, a conversion flow from the Virtex-II Pro FPGA to the Zelix MPGA was presented. The conversion flow checks the timing integrity with respect to the Virtex-II Pro FPGA. Two timing problems in the conversion flow were identified; the first is the necessity to have the same (or better) critical path as in the FPGA; the second is to fix the hold-time violations. The conversion flow consists of: importing the FPGA gate-level netlist, importing the FPGA placement, routing, buffer insertion, physical verification, and layout creation. The routing and buffer insertion tools have been created for the Zelix MPGA. Physical verification uses an interconnect model, and is rather simple. Layout creation is an automated process.

Figure 4.17: Blockage problem

Figure 4.18: Adding metal six to solve the blockage problem

Chapter 5

Sample Design Mappings on Zelix MPGA

5.1 Introduction

This chapter refers to the conversion flow from the Virtex-II Pro FPGA to the Zelix MPGA described in Chapter 4. Because the Zelix MPGA has to be a versatile platform, sample designs cannot cover the whole range of possible applications. In particular, the number of routing tracks per channel, as an architectural parameter, cannot be specified on this basis. However, the investigations may give indications of the range of number of tracks that appears reasonable. Final experience can be obtained only by running a number of real projects in the future.

The first objective of this chapter is to show that timing integrity can be achieved. The second is to determine the associated architectural cost of doing this, principally of finding a 'reasonable' number of routing tracks per channel. The third objective is to compare the delays of an FPGA and an MPGA.

The number of tracks per channel is still a flexible parameter in the Zelix MPGA. As was shown in Chapter 3, this parameter is strongly related to the

final physical area. A further parameter is the drive strength of the cells, particularly, the drive strength and placement density of the clock buffers that have an impact on the timing versatility of the MPGA.

The remaining sections in this chapter are organized as follows:

Section 5.2: Experimental set-up. Section 5.3: Synthesis process. Section 5.4: Virtex-II Pro placement and routing. Section 5.5: Zelix MPGA routing. Section 5.6: Buffer insertion process in the Zelix MPGA. Section 5.7: STA comparison results. Section 5.8: Hold time violations in the Zelix MPGA. Section 5.9: Mask programming of an FIR. Section 5.10: conversion flow time. Section 5.11: summary of the chapter.

5.2 Experimental Preliminaries

Figure 5.1 shows the flow used in my experiments. It has two main components: the Virtex-II Pro FPGA part, and the Zelix MPGA part. In the FPGA part, the Xilinx design flow is used to get the necessary input files for the conversion flow and for the STA analysis. In the Zelix MPGA part, the conversion flow as presented in the last chapter is followed up until the STA analysis.

The FPGA used in the experiments is the Xilinx Virtex-II Pro XC2VP2 device, which has 352 logic tiles arranged in 16 rows and 22 columns.

The benchmark circuits are obtained from different sources: ITC'99 benchmarks circuits [23], OpenCores [55], and Infineon Technologies AG. The ITC'99 circuits consist of 22 sequential circuits, ranging from low to high complexity, with some industrial design subset. The circuits are described in the structural VHDL language (RTL). OpenCores is a public webpage where people post free hardware modules. These hardware modules are a set of circuits described in a structural HDL language (such as VHDL or Verilog) and are written to be mapped onto FPGAs. The Infineon Technologies circuits are a collection of designs from former Master's theses [10] [89] [92] or in one instance from an

Figure 5.1: STA flow used in the experiments

industrial case.

A survey of the functionality and source of the circuits is given in Table 5.1. The application domains of these circuits are digital signal processing and control processing. The complexity of all the circuits is safely matched with the Xilinx Virtex-II Pro XC2VP2 device, the largest circuit utilizing about 50% of its logic resources.

5.3 Synthesis

The circuits are synthesized using Synplicity Synplify Pro 8.5 [82]. The synthesis tool uses an algorithm (named BEST algorithm) that permits various options. The options used in this study are chosen for optimizing the delay by minimizing size. The available options are: maximum fanout, sharing resources, pipelining, retiming, and the use of the FSM explorer. The constraint of low fanout results in replicated logic but obtains a good timing¹ ; maximum

¹For an FPGA, logic replication is less critical than for an ASIC.

Circuit Name	Original Function	Source
ITC_b01	Finite State Machine (FSM) comparing serial flows	ITC'99
ITC_b04	Compute min and max	ITC'99
ITC_b05	Elaborate the contents of a memory	ITC'99
ITC_b11	Scramble string with variable cipher	ITC'99
ITC_b12	1-player game (guess a sequence)	ITC'99
IFX_pp	State Machine of a Protocol Processor	Infineon
IFX_iir	Bi-quad IIR Filter	Infineon
IFX_ace	Associative Search Processor	Infineon
OC_fir	FIR Filter	OpenCores
OC_fft	Fast Fourier Transform	OpenCores

Table 5.1: Benchmarks circuits descriptions

fanout was set to four. With this setting, the synthesis tool occasionally uses a larger fanout when it is better for timing. Resource sharing is used for optimizing area; with this option, the synthesis tool uses the same logic resources for similar operations. As I want to optimize the delay, the resource sharing option is disabled. The pipelining option allows balancing the path delays in the stages of a pipelined datapath, which could be an adder or a multiplier, by shifting logic functionality across the boundaries of the flip-flops. This option allows reduction of the longest stage delay, and thus increases the clock frequency. Hence this option is used in my experiments. I also use the retiming option, which moves registers across the combinational logic to improve the timing. The FSM explorer tries different state machine optimizations, and then chooses the best implementation. As the Zelix MPGA does not have embedded blocks (e.g. SRAM blocks), the synthesis tool is constrained not to use these blocks. The clock, input and outputs of the circuit are not constrained. Their characteristics are: size of the circuit (number of LUTs, FFs, AND

Circuit	#LUTs	#FFs	#Other	#Total	#In Pad	#Out Pad	#I/O
ITC_b01	10	13	0	23	4	2	6
ITC_b04	118	101	61	280	13	8	21
ITC_b05	197	90	53	340	3	36	39
IFX_pp	145	15	247	407	122	17	139
ITC_b11	209	90	193	492	9	6	15
IFX_ace	446	217	26	689	17	48	65
ITC_b12	436	229	84	749	7	7	14
OC_fft	598	604	868	2070	52	40	92
IFX_iir	504	361	1471	2336	34	25	59
OC_fir	657	534	1205	2396	10	10	20

Table 5.2: Characteristics of the test examples

gates...), number of primary inputs and outputs, maximum fanout, average fanout, fanout distribution (number of nodes with fanout 0,1,2, etc.) and number of nets. These characteristics are presented in Table 5.2, and Table 5.3. All the circuits use sequential logic and only one clock. The number of gates is the same in the FPGA and the MPGA, except for the buffers. The characteristics differ in the MPGA because of the different number of buffers. The fourth column of Table 5.2 shows the number of elements like XOR2 gates. We can see that a DSP circuit like a biquad filter (IFX_iir) uses these gates extensively to implement arithmetic functions. The largest number of nets is 2406 (see Table 5.3).

5.4 Virtex-II Pro Placement and Routing

The Virtex-II Pro FPGA is placed and routed with the ISE tools. A Perl-script (Xplorer) is used to optimize the timing [58]. It has been reported that this script provides up to 70% performance improvement. Xplorer performs a timing-driven packing with high effort level. It also tries to explore register duplication in order to improve timing. Xplorer uses timing-driven placement

Circuit	#net	avrg. fanout	max. fanout
ITC_b01	27	1.52	3
ITC_b04	293	2.02	38
ITC_b05	355	2.76	25
IFX_pp	513	1.93	23
ITC_b11	501	2.26	28
IFX_ace	752	2.66	86
ITC_b12	756	2.62	46
OC_fft	2098	2.1	8
IFX_iir	2371	2.17	22
OC_fir	2406	2.08	38

Table 5.3: Net characteristics of the test examples

and routing with high effort level. The timing exploration of this script is mainly based on the packing step.

The largest circuit uses around half of the Virtex-II Pro device area. To test the CAD tools, I used different placement and routing cases. In my experiments the high effort placement means that the circuit is placed in an enclosed region of the array architecture. In a low placement effort, the circuit is placed in the whole array. I restrict the placement area by introducing a constraint file in the Xplorer tool.

The placement discussion applies for both the Virtex-II Pro FPGA and the Zelix MPGA. As the placement is at slice level, the result of high effort placement corresponds with the packing step. So with high placement effort, the number of tiles used is smaller, i.e. there are more slices per tile. The results of Table 5.4 show that, with high placement effort, the number of nets that are used in the top-level interconnect after the routing step is reduced in average by 4.82%. On average, the number of nets inside a tile is increased by

Circuit	Placement Low Effort			Placement High Effort		
	#tiles	Lcl Nets	Glbl Nets	#tiles	Lcl Nets	Glbl Nets
ITC_b01	3	18	9	2	18	9
ITC_b04	30	180	113	16	186	107
ITC_b05	37	157	198	25	166	189
IFX_pp	36	304	209	24	293	220
ITC_b11	40	297	204	29	297	204
IFX_ace	84	384	368	61	405	347
ITC_b12	76	300	456	63	344	412
OC_fft	200	1041	1057	106	1069	1029
IFX_iir	138	1625	746	102	1664	707
OC_fir	162	1557	849	89	1655	751

Table 5.4: Number of tiles, local nets, and global nets after packing/ placement

3.75%. Intuitively, with more local nets the delay is reduced because the total wire-length is smaller. The tile utilization of the OC_fft circuit is 53%, over all the tiles of the device.

5.5 Zelix MPGA Routing

The placement on the FPGA is re-used as input information for the Zelix routing tool. The objective of the routing is minimizing a cost function over all nets of a sample design. In a simple case, for example, the sum of all wire lengths (total wire length) is minimized².

The physical aspect of the routing in the Zelix MPGA is the number of tracks per channel. In general, with rich routing resources (many tracks per channel), a better timing (better performance) is obtained than with limited routing resources. The reason is that routing congestions are less likely. Due to the full programmability of the switch matrix, and the abandonment of hierarchical routing resources, a lower number of tracks per channel is possible compared with the FPGA.

²This is an unspecific approach which does not give special emphasis to critical nets.

It was observed during the experiments that the elapsed time for routing increases when the number of tracks per channel is small, while it decreases with growing numbers of tracks. This gives an indication of the effort of the tool. In the experiments, a high routing effort means that the routing is performed with the minimum possible number of tracks per channel. With low routing effort, the number of tracks per channel is set to 180.

Table 5.5 and Table 5.6 show the number of tracks per channel, and the resulting tile area with different effort cases. HighArea in the routing effort means that the routing is performed in a restricted area of the MPGA array instead of using the whole MPGA device array. It can be observed that the Virtex-II Pro FPGA needs a substantially higher number of tracks per channel than the Zelix MPGA. This is partially because the full population programmability in the switch matrix and I/O crossbar of the Zelix MPGA results in an efficient routing. Moreover, with a hierarchical interconnect, more tracks per channel need to be predefined in the architecture. The use of a hierarchical programmable interconnect in an FPGA is needed to reduce the delay and area [13]. Although we have more top-level nets with low placement effort, the number of tracks per channels is smaller, because with high placement effort the logic is placed in a small area, so it is difficult for the router to find a free track. The area of a tile in the Zelix MPGA is calculated with Equation 3.1. The Virtex-II Pro uses the total tile area number (see Table 2.2 in Chapter 2) in the area reduction comparison. The purpose of this comparison is to determine the benefit of reducing tracks per channel compared with the Virtex-II Pro FPGA. The area ratio of reduction is in the range 71% to 91.57%.

Circuit	Plcmnt Effrt	Rtng Effrt	Zelix #trcks	Virtex-II #trcks	Area μm^2	Area Reduc. %
ITC_b01	Low	Low	x=6 y=2	x=25 y=5	2703	90.99
ITC_b01	Low	High	x=2 y=2	x=25 y=5	2473	91.76
ITC_b01	High	Low	x=7 y=1	x=31 y=2	2732	90.89
ITC_b01	High	High	x=3 y=2	x=31 y=2	2530	91.57
ITC_b01	High	HighArea	x=3 y=2	x=31 y=2	2530	91.57
ITC_b04	Low	Low	x=12 y=20	x=44 y=38	3656	87.81
ITC_b04	Low	High	x=7 y=6	x=44 y=38	2887	90.38
ITC_b04	High	Low	x=19 y=20	x=71 y=40	4151	86.16
ITC_b04	High	High	x=11 y=10	x=71 y=40	3264	89.12
ITC_b04	High	HighArea	x=13 y=13	x=71 y=40	3494	88.35
ITC_b05	Low	Low	x=22 y=30	x=85 y=38	4731	84.23
ITC_b05	Low	High	x=11 y=11	x=85 y=38	3297	89.01
ITC_b05	High	Low	x=37 y=23	x=79 y=32	5690	81.03
ITC_b05	High	High	x=14 y=12	x=79 y=32	3538	88.24
ITC_b05	High	HighArea	x=19 y=18	x=79 y=32	4081	86.40
IFX_pp	Low	Low	x=19 y=16	x=57 y=35	4011	86.63
IFX_pp	Low	High	x=9 y=9	x=57 y=35	3105	89.65
IFX_pp	High	Low	x=22 y=15	x=54 y=37	4193	86.02
IFX_pp	High	High	x=11 y=11	x=54 y=37	3297	89.01
IFX_pp	High	HighArea	x=11 y=11	x=54 y=37	3297	89.01
ITC_b11	Low	Low	x=22 y=34	x=65 y=58	4874	83.75
ITC_b11	Low	High	x=13 y=12	x=65 y=58	3461	88.46
ITC_b11	High	Low	x=30 y=35	x=87 y=49	5568	81.44
ITC_b11	High	High	x=14 y=14	x=87 y=49	3594	88.02
ITC_b11	High	HighArea	x=21 y=21	x=87 y=49	4333	86.56
IFX_ace	Low	Low	x=22 y=20	x=67 y=35	4372	85.42
IFX_ace	Low	High	x=9 y=9	x=67 y=35	3105	89.65
IFX_ace	High	Low	x=36 y=38	x=104 y=37	6209	79.30
IFX_ace	High	High	x=19 y=19	x=104 y=37	4116	86.28
IFX_ace	High	HighArea	x=22 y=20	x=104 y=37	4372	85.42

Table 5.5: Number of tracks per channels, and area. Plcmnt Effrt: Placement Effort, Rtng Effrt: Routing Effort, #trcks: number of tracks per channel, Reduc.: Reduction.

Circuit	Plcmnt Effrt	Rtng Effrt	Zelix #trcks	Virtex-II #trcks	Area μm^2	Area Reduc. %
ITC_b12	Low	Low	x=33 y=24	x=82 y=45	5392	82.03
ITC_b12	Low	High	x=14 y=13	x=82 y=45	3561	88.13
ITC_b12	High	Low	x=56 y=40	x=119 y=76	8223	72.59
ITC_b12	High	High	x=22 y=21	x=119 y=76	4408	85.31
ITC_b12	High	HighArea	x=28 y=27	x=119 y=76	5008	83.01
OC_fft	Low	Low	x=25 y=31	x=63 y=40	5005	83.32
OC_fft	Low	High	x=12 y=11	x=63 y=40	3362	88.79
OC_fft	High	Low	x=48 y=69	x=80 y=90	8700	71.00
OC_fft	High	High	x=30 y=30	x=80 y=90	5376	82.08
OC_fft	High	HighArea	x=38 y=38	x=80 y=90	6390	78.70
IFX_iir	Low	Low	x=24 y=24	x=85 y=39	4669	84.44
IFX_iir	Low	High	x=13 y=13	x=85 y=39	3494	88.35
IFX_iir	High	Low	x=40 y=33	x=87 y=56	6365	78.78
IFX_iir	High	High	x=17 y=18	x=87 y=56	3938	86.87
IFX_iir	High	HighArea	x=21 y=21	x=87 y=56	4333	85.56
OC_fir	Low	Low	x=29 y=27	x=72 y=41	5179	82.73
OC_fir	Low	High	x=13 y=12	x=72 y=41	3461	88.46
OC_fir	High	Low	x=62 y=54	x=137 y=111	9533	68.22
OC_fir	High	High	x=31 y=30	x=137 y=111	5460	81.80
OC_fir	High	HighArea	x=40 y=39	x=137 y=111	6614	77.95

Table 5.6: Number of tracks per channels, and area CONT'

5.6 Zelix MPGA Buffer Insertion

After the routing, timing fixes are made by inserting buffers. The benefit of buffer insertion for signal delay is that a weak signal slope is refreshed, while the improvement is larger than the intrinsic delay that is added by the buffer. For instance, improving a slope from 3 ns to 1 ns at the cost of a buffer delay of 1 ns results in a total timing improvement of 1 ns. In my architecture I used one buffer type with an intrinsic delay of 43 ps. The intrinsic delay is needed to use the buffer delay model of equation .3 of chapter 4. The intrinsic delay

is the delay of the cell with a zero load capacitance. Figure 5.2 shows the delay behavior of the buffer. The values given are from the library data book. For purpose of buffer insertion the path delays of all signals are analyzed. Paths that exceeded the timing budget given from the original FPGA implementation were buffered. The Zelix architecture provides buffer resources for each one-length segment in both directions (two buffers per segment). In my conversion methodology, the buffer insertion algorithm is applied to all the nets. The number of inserted buffers is shown in Table 5.7. The case with the lower number of routing tracks (LowLow, and HighLow) needs more buffers, because the average wire-length per net is larger. Assuming 30 tracks per channel, a tile in the Zelix- MPGA has 120 buffers. So, the buffer utilization per tile (the number of buffers used per available buffers in a tile) is less than 7%. One way in which the reduction of buffers reduces the final area of a tile is by using more metallization layers for the top-level interconnect. A comparison with the number of buffers in the clock network (see Table 5.8) shows that the number of buffers in the clock network is of the same order as the buffers inserted in the circuit nets. The high effort placement uses fewer clock buffers, because the clock network applies in fewer quadrants, and the register packing is higher. The total number of buffers is shown in Table 5.9.

5.7 STA Comparison

To verify that the timing after the conversion matches the timing of the FPGA, STA has to be applied. This is because the FPGA design is purely digital, requiring a strict synchronous design methodology. By means of a script, all the logic paths are compared one-by-one with the Virtex-II Pro. Table 5.10 shows the results. As we can see, without buffer insertion there are paths that have longer delays compared with the Virtex-II Pro. Buffer insertion affects

Figure 5.2: Linear buffer delay.

signal refresh, in the sense that the improvement of the signal slope exceeds the additional intrinsic delay of the buffer. With rich routing resources, low routing effort, and buffer insertion, the logic paths in the Zelix MPGA always have smaller delay. This is because, with rich routing resources, we are likely to have a wire-length that is shorter than in the FPGA. As a one-segment wire in the Zelix MPGA has smaller delay, due to the shrinking factor and the mask programmability, the delays are smaller. With reduced routing resources, some wired paths are larger than in the Virtex-II Pro. The number of logic paths with larger delay is small compared to the total number.

In Figure 5.3 the total area for each circuit is calculated using the area equation 3.1 for a tile, and multiplying the result by the number of tiles. This calculation is not accurate, because in the relaxed routing there are more empty tiles in the place area. But, with this total area we can get an idea of the area cost. We see that the best area efficiency is reached with the placement and routing constrained. The problem is that with the constrained routing, the

Circuit	Inserted Buffers.				
	Plc/Rt	Plc/Rt	Plc/Rt	Plc/Rt	Plc/Rt
	LowLow	LowHigh	HighLow	HighHigh	HighHighArea
ITC_b01	3	4	3	4	4
ITC_b04	94	89	74	86	101
ITC_b05	186	191	188	194	190
IFX_pp	77	81	64	71	67
ITC_b11	194	206	192	216	193
IFX_ace	504	514	506	521	510
ITC_b12	384	407	457	469	465
OC_fft	377	378	459	532	459
IFX_iir	358	365	414	415	429
OC_fir	495	508	532	552	576
Buffers per Tile					
ITC_b01	1.00	1.33	1.5	2.00	2.00
ITC_b04	3.13	2.96	4.62	6.31	6.31
ITC_b05	5.03	5.16	7.52	7.76	7.60
IFX_pp	2.14	2.25	2.67	2.97	2.79
ITC_b11	4.85	5.15	6.62	7.45	6.65
IFX_ace	6.00	6.12	8.30	8.54	8.36
ITC_b12	5.05	5.36	7.25	7.44	7.38
OC_fft	1.89	1.89	4.33	5.01	4.33
IFX_iir	2.59	2.64	4.06	4.07	4.21
OC_fir	3.06	3.14	5.98	6.00	6.47

Table 5.7: Inserted buffers. Plc: placement effort, Rt: routing effort, High-Area: High routing effort with restricted area.

delay is degraded. One solution is the possibility in the MPGAs to use more metal layers for the routing, without increasing the total area. The problem with more metallization is the increase of the device cost, and that buffer insertion becomes more difficult in the upper layers.

The FPGA has cases with better average delay without buffer insertion (Figure 5.4). After buffer insertion the Zelix MPGA shows an improvement of the average delay from 28% to 78% (Figure 5.5). This is because the numbers of paths with larger delays are not significant on average. The effort cases are of the same order in average delay.

If we compare the critical path results without buffer insertion (Figure 5.6),

Circuit	Clock Buffers	
	Low	High
ITC_b01	12	11
ITC_b04	84	67
ITC_b05	78	75
IFX_pp	25	18
ITC_b11	95	84
IFX_ace	240	209
ITC_b12	213	191
OC_fft	565	427
IFX_iir	347	295
OC_fir	459	360

Table 5.8: Buffers in the clock network. Low: Low placement effort, High: High placement effort.

we see that, when the routing is constrained, the Zelix MPGA does not always have a better critical path. In our experiments, with buffer insertion the critical path is improved by 48.32% in average. As the ITC_b01 is small, the average is calculated without it. The critical path is always compared with the same placement in the Virtex-II Pro, and the Zelix MPGA. As shown in Figure 5.7, the Zelix MPGA has a better critical path after buffer insertion.

5.8 Zelix MPGA: Hold Time

Hold-time violations are checked by comparing the fast logic path with the clock delay by means of a Perl-script. The library cells are used with the best case corner. The interconnect delays uses the model described in Section 4.3.6, i.e. there is no best or worst case condition. In Section 4.2, I stated that the Zelix MPGA has no hold time violations because of its physical design. Experimental results confirm this assertion.

It is fair to remember that, in this thesis (and therefore with the experimental results), it is assumed that there is only one clock domain. With multiple clock

Circuit	Total Buffers				
	Plc/Rt	Plc/Rt	Plc/Rt	Plc/Rt	Plc/Rt
	LowLow	LowHigh	HighLow	HighHigh	HighHighArea
ITC_b01	15	16	14	15	15
ITC_b04	178	173	141	168	168
ITC_b05	264	269	263	269	265
IFX_pp	102	106	82	89	85
ITC_b11	289	301	276	300	277
IFX_ace	744	754	715	730	719
ITC_b12	597	620	648	660	656
OC_fft	942	943	886	959	886
IFX_iir	705	712	709	710	724
OC_fir	954	967	892	894	936

Table 5.9: Total number of buffers. Plc: placement effort, Rt: routing effort, HighArea: High routing effort with restricted area.

domains, hold- time violations can appear.

Figure 5.8 shows the ITC_b11 case. As we can see, the hold-time slack is always positive, i.e. the clock delay and the flip-flop hold time is always shorter than the path delay (see equation 4.3).

5.9 Zelix Mask-Programming

The final step of the conversion flow is layout creation. As an example, the layout of the FIR filter with low placement and low routing effort has been created. Figure 5.9 shows the mapping of this circuit onto a 16 x 22 array of tiles. The outer ring shown in the figure is to facilitate the routing between the logic core, and the I/O pads. We can see from the figure that the low placement distributes the logic around the inner device. A detailed view of part of the circuit is given in Figure 5.10. We can see that the routing and the power grid use only metal 3 and metal 4.

Circuit	without buffer insertion					Logic Paths
	LowLow	LowHigh	HighLow	HighHigh	HighHighArea	
ITC_b01	0	0	0	0	0	15
ITC_b04	54	55	55	55	55	109
ITC_b05	10	25	10	54	15	126
IFX_pp	0	0	0	0	0	32
ITC_b11	26	65	19	50	19	96
IFX_ace	5	34	7	18	8	225
ITC_b12	156	160	161	164	155	235
OC_fft	11	19	11	0	31	644
IFX_iir	42	59	52	83	55	386
OC_fir	118	144	124	195	147	544
with buffer insertion						
ITC_b01	0	0	0	0	0	15
ITC_b04	0	1	0	3	1	109
ITC_b05	0	0	0	1	0	126
IFX_pp	0	0	0	0	0	32
ITC_b11	0	15	0	9	1	96
IFX_ace	0	0	0	0	0	225
ITC_b12	0	2	0	1	1	235
OC_fft	0	0	0	13	4	644
IFX_iir	0	1	0	6	1	386
OC_fir	0	0	0	1	0	544

Table 5.10: Data paths with larger delay than Virtex-II Pro

5.10 Conversion Flow Time

The flow was run in a Unix machine with the SunOS 5.8 operating system. The workstation has four Sun-Fire-480s CPUs running at 900MHz with 16384MB. For Linux, the RedHat 7.2 operating system was used. The tools were run on a PC with 2 AMD Athlon(TM) CPUs at 1733 MHz with 3648MB.

Table 5.11 shows the results of the total time conversion flow. The total time for the conversion was 77 minutes.

5.11 Summary: Delay

The experimental results of the conversion flow have been presented. Ten sample designs were mapped through the conversion flow. The effect of placement and routing was studied. It could be observed that, with reduced routing resources and high routing effort, the average path delays are higher than with rich routing resources. However, timing integrity in terms of set-up correctness could be satisfied, which was the goal of the conversion. Hold-time violations

Figure 5.3: Total area

were not observed. In comparison with the Virtex-II Pro, the average of the critical path is improved 48% with the Zelix MPGA. As the Zelix MPGA is a replacement for an FPGA, there is no direct profit with this delay improvement.

Figure 5.4: Average delay without buffer insertion

Figure 5.5: Average delay with buffer insertion

Figure 5.6: Critical path delay without buffer insertion

Figure 5.7: Critical path delay with buffer insertion

Figure 5.8: Hold time slack of the ITC_b11 circuit

Figure 5.9: Layout of the FIR circuit

Figure 5.10: Detail of the FIR circuit layout

Flow Step	FFT	IIR	FIR	Average
Synthesis	132	120	85	112
Place&Route FPGA	634	550	493	559
Total FPGA	766	670	578	671
Netlist Mod.	110	115	120	115
Placement	930	980	900	937
Routing	10	10	10	10
Prep. data buffer insertion	1820	1630	1700	1717
Buffer Insertion	635	595	680	637
Parasitic Model	1320	1150	1200	1223
STA Analysis	15	19	17	17
Total Conversion	4840	4499	4627	4655
Mask Programming	N.A.	N.A.	11210	N.A.
Total Conv. w/mask prog.	N.A.	N.A.	15837	N.A.

Table 5.11: Conversion flow time (seconds)

Chapter 6

Zelix MPGA Power Consumption Flow

6.1 Introduction

Due to reliability issues, battery life, and packaging costs, power consumption has become a critical design issue in some applications [67]. To detect excess power demands, they have to be estimated and evaluated during VLSI design. Power estimation is carried out at different stages of the design process. A spreadsheet can be used at the most abstract level, where only an approximation of the number of transistors and their frequency are available (e.g. [6]), while, after synthesis, gate-level power simulators can be used (e.g. [71]); at this level, a load model is needed to estimate the parasitic capacitances. With physical implementation decided, we can perform transistor-level simulations; knowing the layout allows the extraction of the interconnect capacitances; finally, once we have the device fabricated, we can directly measure the power consumption. Each of the lower refinement stages provides more power estimation information, increasing the accuracy of the more abstract level tools; for instance, after synthesis we can use a spreadsheet estimation, knowing the exact number of transistors. With deep-submicron technologies the amount of power consumed by interconnect parasitic capacitances becomes significant,

leading to poor gate-power simulations when a load model is used. In some cases, a spreadsheet tool has the same precision as a gate-power simulator. A good prediction of the capacitances of the wires is needed. Compared with ASICs, FPGAs have a programmable prefabricated routing architecture; the interconnect capacitances can be estimated without parasitic extraction. FPGA vendors exploit this feature to provide accurate power estimation tools [7] [98], gate-power simulators yielding good capacitance estimations. Degalahal and Tuan [27] report a power consumption average estimation error of 18%. Power estimation in MPGAs suffer from the same problem as ASICs: poor prediction of interconnect capacitances.

This chapter presents a power estimation methodology for the Zelix MPGA. As the Zelix MPGA has a pre-defined interconnect structure, the capacitance calculation is straightforward. The estimation methodology uses a gate-power simulation tool.

For estimating power consumption in the Zelix MPGA, two key elements are necessary: a gate library, and a parasitic interconnect model. The library describes the power behavior of the cells and the interconnect model is necessary for fast estimation of the load capacitance.

The rest of the sections of this chapter is organized as follows:

Section 6.2: Review of the basic concepts of power consumption. Section 6.3: Description of a power estimation tool of the Virtex-II Pro FPGA. Section 6.4: Power methodology. Section 6.5: Flow validation with transistor-level simulations. Section 6.6: Experimental results with the proposed methodology. Section 6.7: Chapter summary.

6.2 Power Consumption in CMOS Circuits

The power consumption in CMOS circuits has two main contributions: dynamic and static power consumption. Dynamic power consumption is due to

the activity of the circuit. It is divided into load power consumption and short-circuit power consumption. The first is in the wires of the circuit, the second in the logic. It is well known that the power dissipated due to the charge and discharge of the output capacitance C_{load} of a gate is:

$$P_{load} = C_{load} V_{DD}^2 f_p \quad (6.1)$$

where V_{DD} is the supply voltage, and f_p is the output-gate switching frequency. The f_p term is usually expressed as:

$$f_p = \frac{\alpha}{2} f_{clock} \quad (6.2)$$

where α is the switching of a net, and f_{clock} is the clock frequency. Short-circuit power consumption takes place when a current path from the power supply V_{DD} to the ground V_{SS} exists. This phenomenon occurs when the input signal of a gate changes, when both the NMOS and PMOS transistors in CMOS circuits conduct simultaneously for a short period of time causing a current flow from V_{DD} to V_{SS} . The static power consumption occurs as a result of the leakage current in the transistors. The leakage current is caused either by a non-zero current between the source and the drain transistor terminals when the MOS transistor is switched off (subthreshold leakage), the reverse current of diode, or by the tunneling current through the gate oxide of an MOS transistor (gate leakage). The static power depends on the CMOS technology parameters, and the temperature at which the circuit works.

6.3 Power Estimation on Virtex-II Pro FPGA

To automate the power estimation process, Xilinx includes an additional tool (called Xpower) with their CAD tools [98]. The Xpower tool is a power-gate tool to estimate power consumption in early stages of the design, based on the routing information, the gate-level netlist and the switching activity information. The routing information includes the interconnect usage. As the

Virtex-II Pro has a finite hierarchical interconnect, the Xpower tool estimate the interconnect capacity by multiplying the number of resources used in one net by the wire resource. For example, a net with two double lines has a total capacity of twice the capacity of a double line. The capacity of a line is pre-characterized by power consumption measurements on the silicon device [72]. The gate-level netlist is generated after the technology synthesis. Given this gate-level netlist, it is possible to simulate the design; if the input stimuli of this simulation are correctly generated, the switching activities inside the circuit can be calculated.

Experimental results with this tool show that the major contributor of power consumption in the Virtex-II (a CMOS 150nm technology device) is the interconnect. The clock takes around 20% of the total dynamic power consumption. With the Spartan-III (CMOS 90nm technology device), the clock contributes about 39% of the total power, and the interconnect contributes about 45% [27].

6.4 Zelix MPGA Power Estimation Methodology

I propose a similar technique to Xilinx Xpower to estimate the power consumption on the Zelix MPGA. Figure 6.1 shows the methodology used to study the power consumption. The flow starts with a gate-level netlist, and a routing file; the gate-level netlist includes elements such as LUTs, and clock buffers. The routing file is the circuit routed onto an architectural abstraction of the MPGA interconnect. The routing file is sufficient, because the power estimation flow does not depend on the physical layout. With the routing file, the net capacitances are calculated, generating a capacitance file in SPEF format (an IEEE standard for representing parasitic data of wires in a chip). The

Figure 6.1: Power estimation flow

capacitances are modeled using the technology process data-book. A gate-level simulator [50] is used to get the net toggling frequencies. As we have complex cells like LUTs, it is necessary to create a library with the gate function description for the gate-simulator. With the gate-level netlist, the net activity, and net annotated capacitances file, we can simulate the power consumption with the PowerTheater tool [71]. PowerTheater is a comprehensive set of power tools to improve power efficiency for cell-based designs, which focuses on the architectural, RTL, and gate levels. PowerTheater is used with the gate-level modus. The power simulator tool uses a logic library that lists the power of cells, depending on their input and output-gate toggling, and capacitances. The standard cells' power characteristics are imported to our library from the technology vendor. The LUT power values are derived from transistor-level simulations. The detailed report includes the logic, clock network and load power consumption, noting that the logic power includes the internal interconnect power. For example, if we have a multiplexer-cell that consists of a pass-gate, and inverters in the inputs and outputs, the power of

the internal interconnect between the pass-gate, and the inverters are reported as logic power consumption. The net frequencies are also reported.

I estimate the power consumption only for the logic core of the MPGA; the power of complex blocks, such as SRAM memories or DSP processors, is usually estimated separately, when integrating the estimations to give the final power result.

6.5 Power Consumption Estimation Flow Validation

To validate the power flow, transistor-level simulations have been performed. Transistor-level netlists, without parasitic capacitance annotations, of four circuits have been created. Simulations have been performed with Synopsys NanoSim [80]. Figure 6.2 shows the results. We can see an average difference of 20% with respect to the power flow; the maximum difference is 31% with the ITC_b05 circuit. The power results of the transistor-level simulations are always lower than those from the gate-level simulations. The latter is caused in part because the net capacitances are not annotated in the transistor-level netlist. Note that the load in this transistor-level simulation is the input-gate capacitances. The simulation without net capacitances is useful to see the power consumption of the gates. This kind of simulation has been used to adjust some FPGA power models (see [47]). If we annotate the load power consumption in the transistor-level simulation results (Nanosim+PoweTheater_load in Figure 6.2), there is an average difference of 4.83%. There is no large simulation time difference between a transistor-level or gate-level tool. Both tools produce results in less than 10 minutes. For example, the ITC_b05 has a run time of 313 seconds with transistor-level simulations, and 152 seconds with gate-level simulations. The benefit of using a

Figure 6.2: Comparison between transistor-level simulation (NanoSim, and Nanosim+PoweTheater_load), and proposed power consumption estimation methodology (PowerTheater).

gate-level tool is that the tool provides a detailed report in which the sources of power consumption can be seen in detail. A detailed report in a transistor-level simulator is rather complex. For instance, if we want to know the power consumption of the clock network, the transistor-level tool requires an isolation of the clock network in the transistor netlist. Another difficulty arises in the description of the vector stimuli. The gate-level tool requires stimuli to be described in verilog, whereas the transistor level tool requires binary-vector stimuli.

6.6 Power Consumption Flow Results

This section presents the experimental results of ten benchmark circuits proceeding through the power flow described in Section 6.4. The circuit characteristics can be found in Chapter 5. Each circuit has been synthesized and converted from a Virtex-II Pro FPGA, using the Zelix MPGA conversion flow. The placement and routing has been performed with high effort. The Virtex-II

Pro device used is 2vpfg256-7.

The input stimuli of the gate-simulator are randomly generated, apart from the clock, and the control signals such as reset. The frequency of the clock is set to 100MHz. The data input signals are always a multiple of the clock period. The simulation time for the gate-simulator has been set from 3000 to 10000ns depending on the output signals. An output signal value is toggled a minimum of ten times in our experiments.

The Xilinx Xpower tool [98] is used to estimate the power of the Xilinx Virtex-II Pro. Comparisons with the Xilinx Virtex-II Pro are done with the same input stimuli, and simulation time as for the Zelix MPGA.

6.6.1 Dynamic Power Consumption

The results of the dynamic power consumption are reported in Table 6.1, and the average over all the circuits is shown in Figure 6.3. The major contributor to the dynamic power consumption is the clock network, partly because around 80% of the nets have a switching frequency below 20MHz. Figure 6.4 shows the nets' toggling probabilities of the ITC_b05 circuit¹. Figure 6.5 shows the nets' toggling probabilities of the OC_fir circuit. In the Virtex-II Pro FPGA, the clock power consumption is close to the interconnect power consumption. The power of the interconnect in the Zelix MPGA is smallest because of the smaller wire-length, the lack of use of programmable switches, the smaller number of buffers, and a more efficient routing caused by the full programmability in the input/output crossbar and switch matrix. A 1-segment wire in the Zelix MPGA is reduced by 23% in the vertical direction, and 75% in the horizontal direction when compared with the Virtex-II Pro FPGA. For example, the final routing of the OC_fir has 9504 1-segment wires in the Zelix MPGA and 37913 1-segment wires in the Virtex-II Pro FPGA. The numbers of buffers used in the

¹The meaning of static probability is the same as the toggling probability; see as an example [9].

Circuit	Zelix MPGA (mW)				Virtex-II Pro FPGA (mW)				Reduction (%)
	Logic	Inter.	Clock	Total	Logic	Inter.	Clock	Total	
ITC_b01	0.19	0.12	0.26	0.57	0.63	1.23	2.64	4.50	87.39
ITC_b04	0.91	0.11	1.37	2.39	1.74	3.82	2.73	8.28	71.13
IFX_b05	2.28	1.94	0.21	4.43	7.32	18.36	4.04	29.72	85.09
IFX_pp	0.22	0.02	0.36	0.60	0.54	1.65	2.42	4.61	86.98
ITC_b11	1.87	0.51	1.69	4.07	5.78	12.97	4.51	23.26	82.52
IFX_ace	1.80	0.65	4.20	6.65	6.18	16.50	5.43	28.11	76.34
ITC_b12	2.11	0.82	3.68	6.79	9.79	20.80	5.46	36.05	81.17
OC_fft	5.23	1.17	8.69	15.09	10.62	10.08	9.03	29.73	49.24
IFX_iir	1.43	0.33	6.24	8.00	1.53	5.06	8.12	14.71	45.62
OC_fir	3.69	0.68	7.34	11.70	7.39	18.31	8.35	34.05	65.63
Average(%)	34.82	12.49	52.69	100	21.30	46.54	32.16	100	73.11

Table 6.1: Dynamic power consumption. Interconnect (Inter.) includes the buffers power dissipation.

Figure 6.3: Average dynamic power dissipation distribution.

Virtex-II Pro FPGA is 10233, compared with 415 buffers in the Zelix MPGA. Note that the FPGA buffers have more strength (and therefore more power) than the MPGA, because they have to drive the programmable switches. On average, the dynamic power in the Zelix MPGA is reduced by 71.52%².

Figure 6.6 shows the dynamic power dependence of the circuit OC_fir on the clock frequency. The power in the Virtex-II Pro FPGA grows faster than the Zelix MPGA.

²Average without the small circuit ITC_b01.

Figure 6.4: Circuit ITC_b05 histogram of static probability. The number in brackets indicates the range of probability.

6.6.2 Static Power Consumption

Table 6.2 reports the static power consumption. The major contributor to static power is the logic. As the OC_fft has a larger number of buffers in the clock network, it has a larger clock network leakage. The static power consumption represents less than 1% of the total power consumption (see Table 6.2). The Virtex-II Pro FPGA always has a constant static power consumption of 30mW, but this includes the embedded blocks such as SRAMs. Therefore, I do not provide a comparison with the Zelix MPGA.

6.7 Summary: Power Consumption

In this chapter, a power consumption estimation flow for the Zelix MPGA was presented. The flow estimates the power consumption with gate-level information. A comparison with transistor-level simulations shows an average difference of 4.83%. The experimental results show that, on average, around

Figure 6.5: Circuit OC_fir histogram of static probability. The number in brackets indicates the range of probability.

half of the power consumption is in the clock network. In comparison with the Virtex-II Pro, the dynamic power consumption is reduced by 72%. Static power consumption represents less than 1% in the Zelix MPGA.

Figure 6.6: Dynamic power dependence with the clock frequency of the OC_fir circuit

Circuit	Static Power (μ W)			Total Power Consumption (μ W)	Static Power Percentage (%)
	Logic	Clock	Total		
ITC_b01	0.08	0.03	0.11	567	0.02
ITC_b04	1.07	0.16	1.23	2393	0.05
IFX_b05	1.79	0.18	1.97	4433	0.04
IFX_pp	1.29	0.04	1.33	602	0.22
IFX_b11	2.02	0.20	2.22	4068	0.05
IFX_ace	4.20	0.50	4.70	6655	0.07
IFX_b12	4.07	0.46	4.53	6794	0.07
OC_fft	6.21	1.04	7.25	15097	0.05
IFX_iir	5.36	0.75	6.11	8005	0.08
OC_fir	6.78	0.87	7.65	11712	0.07

Table 6.2: Static power consumption on Zelix MPGA

Chapter 7

Conclusion

The flexibility offered by off-the-shelf FPGAs allows for a fast time to market of new and individual systems. Also, the simplified design flow lowers the risks of system design. For low volume quantities, FPGA-based systems require significant lower investment than ASICs, as long as power consumption and physical dimensions do not limit this solution.

MPGAs can be seen as a good and complementary alternative to FPGAs, for addressing the cost and power consumption disadvantages of FPGAs. The key benefit is the ability to prototype in an FPGA, and then to automatically translate the design into an MPGA. The resulting system is nearly as optimized as with ASICs, but at a very fast turn-around time and low risks. This is a new way of system development; on the one hand there is a simplified flow with the possibility of a cheap verification of the prototype, but on the other there is less potential for optimization to meet stringent power budgets. Performance seems to be less critical since FPGAs are very fast today, and the concept inherently offers lot of parallelism.

In the course of this dissertation I reused the cell structure and placement information from the FPGA for the MPGA, while making a new routing for it. This proved to be very efficient with respect to the flow, and the mapped designs could adequately achieve the same timing and performance as reached by the FPGA. Power could be reduced due to shorter interconnect length, and

optimized utilization of cells.

In this dissertation two new LUT-based MPGAs were presented. The first MPGA, r-MPGA, has the same architecture as an FPGA, and preserves the hierarchical interconnect segmentation. The r-MPGA uses the simple concept of substituting the programmable switches of an FPGA by metallization, though it has been shown that this simple concept has a high penalty in terms of area. The second MPGA, Zelix MPGA, has an improved interconnect scheme; the interconnect is planned as a unified one-length segmentation, moreover, the switch matrix can have up to 100% population. Both the fine-segmented interconnect, and the high usability of the switch matrix allowed the routing resources to be reduced in comparison with the FPGA. This contributed to reduced die area and reduced power consumption of the MPGA. However, there is no analytical means to calculate the factor of resource reduction because the nature of the problem is statistical. My decision to reduce the tracks per channel by a factor of five to six compared to the FPGA showed no problems in the course of my investigations.

An automated conversion flow for the Zelix MPGA was presented. The conversion flow has a novel MPGA router, and a specific buffer insertion tool for the Zelix MPGA. Compared to the complicated routing in ASICs, the simple interconnect scheme has the advantage of having an interconnect model that allows prediction of the associated parasitics before layout creation. Accurate STA analysis can be realized without the final layout, thus saving verification time. The experimental results of the conversion flow showed no hold-time violations. In my experiments, the timing integrity was always satisfied after buffer insertion.

The minimum required height of the metallization stack was four layers for the signal routing. Even if additional metallization layers would be considered, for a very robust power distribution, this is a significantly simpler technology than

in FPGAs that use nine or more layers. The number of mask steps for customization of the master circuitry was two metallizations, and two vias. To preserve the aspect ratio of the total chip, embedded macros, most notably the SRAMs, have to be reduced in the MPGA.

Estimations of power consumption showed that static power could be minimized by physically disconnecting unused portions of the master circuitry. My experiments revealed that the Zelix MPGAs consumes 15mW at a size of 200 logic tiles. This indicates that MPGAs also have potential to be used in such application scenarios as portable devices. However, highly miniaturized hand-held applications are unlikely. This low power consumption has also the advantage of a cheap package¹.

Table 7.1, and Table 7.2 summarize the experimental results of this thesis. Altera Stratix HardCopy MPGA can be used as a reference (a point-by-point comparison is not possible because the lack of information). The Zelix MPGA is slightly better in terms of area, delay, and power consumption than the Stratix HardCopy. However, the Zelix MPGA has fewer metallization layers. Stratix HardCopy needs more programmable metal layers because the inefficiency of its routing. The high number of fixed masks is caused mainly because the embedded blocks (e.g. an SRAM block requires five metal layers), and re-use of the Stratix FPGA layout. The impressive difference of the conversion time is mainly because of the physical verification. Meanwhile the Statix Hard-Copy needs a long parasitic extraction time from layout, in contrast to the Zelix MPGA, which does an estimation of the parasitics before layout.

Although this thesis includes no comparisons with other structures such as ASICs or cell-based MPGAs, it is likely that the Zelix MPGA has a drawback in terms of area. ASICs can map circuits more efficiently, but this implies

¹One Watt is the maximum allowed power consumption of a cheap plastic package [88], page 337.

	Zelix MPGA difference	Stratix HardCopy
Area (logic)	82%	N.A.
Area (Device)	58-64%	50-70%
Critical Path	48%	50%
Power Consumption	72%	40%

Table 7.1: Comparison between Zelix MPGA, and Xilinx Virtex-II Pro FPGA, with Altera Stratix HardCopy MPGA as a reference [3]. Percentage indicates reduction with respect to an FPGA.

	Zelix MPGA	Virtex-II Pro	Stratix HardCopy
Conversion Time	1-2 days	0	14 days
Prog. Masks	2	0	3
Total Masks	4	9	7

Table 7.2: Conversion time, and number of masks. Altera numbers from [16].

re-synthesis, placement, etc. Therefore, the advantage of a short turn-around time will always be on the side of the proposed type of MPGA.

In a final perspective, MPGAs suffer from the same disadvantage as gate arrays, namely the predefinition of master dies. Unlike gate arrays, the MPGA has the advantage of FPGA prototyping and very simple conversion. Also, the number of customization masks is reduced. The proposed MPGAs are not likely to be used for high volume systems; their strength is for platform systems that require application- specific adaptation. The risk and cost of mask-changes can be reduced significantly due to FPGA usage. Without precise quantification, in my opinion the total cost advantage of simple design methodology, low risks, and the methodology of mass production open up a major potential for MPGAs in the domain of individualized systems.

7.1 Future Work

This work is to my knowledge the first published research in this area, and has laid the foundation for further research into LUT-based MPGAs.

As the main aim of LUT-based MPGAs is to minimize the high cost of FPGAs, the reduction of the physical area has to be addressed further, since the portion of the logic cells area can be reduced by trading-off number of tracks for masks. Also full- custom design of a tile can reduce the area.

Certainly, further investigation would be needed to investigate the whole concept. To complete the timing investigation at the SRAM interfaces, the design of a full chip would be needed.

Most of all, the number of mapped designs needs to be increased significantly to get statistical experience about their routability with reduced resources.

Bibliography

- [1] “Rapidchip technology fast custom through platform-based design,” *White Paper, LSI Inc.*, 2004.
- [2] C. J. Alpert and A. Devgan, “Wire segmenting for improved buffer insertion,” *Proceedings of the IEEE/ACM Design Automation Conference*, pp. 649–654, June 1997.
- [3] *HardCopy Series Handbook (h5v1)*, Altera Inc., 2005.
- [4] *HardCopy Series Handbook, vol.1, ch. 22: Back-end Timing closure for HardCopy Series Devices (H51013-21)*, Altera Inc., 2005.
- [5] *QuartusII Handbook (QII53006-5.1.0)*, Altera Inc., 2005.
- [6] *QuartusII Handbook, vol.3, ch. 7: PowerPlay Early Power Estimator (QII53006-5.1.0)*, Altera Inc., 2005.
- [7] *QuartusII Handbook, vol.3, ch. 8: PowerPlay Power Analyzer (QII53006-5.1.0)*, Altera Inc., 2005.
- [8] (2006) The Altera website. [Online]. Available: <http://www.altera.com/>
- [9] J. H. Anderson and F. N. Najm, “Active leakage power optimization for FPGAs,” *IEEE Transactions on Computed-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 423–437, 2006.

- [10] A. Beato, “Embedded microprocessor platform supporting compressed code,” Master’s thesis, Polytechnics University of Marche, Marche, Italy, 2003.
- [11] J. Berthold, Infineon Technologies AG, private communication, 2005.
- [12] V. Betz. (1998) EasyGL an easy-to-use graphics interface to X11, PostScript, and Win32 displays. [Online]. Available: <http://www.eecg.toronto.edu/vaughn/easygl/easygl.html>
- [13] V. Betz and J. Rose, “FPGA routing architecture: Segmentation and buffering to optimize speed and density,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, 1999, pp. 59–68.
- [14] —, “Automatic generation of FPGA routing architectures from high-level descriptions,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, 2000, pp. 175–184.
- [15] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer Academic Publishers, 1999.
- [16] J.-C. Bouzigues. (2003) Altera hardcopy devices. presentation at the 2003 NASA MAPLD International Conference. [Online]. Available: <http://klabs.org/richcontent/MAPLDCon03/MAPLDCon03.html>
- [17] S. D. Brown, J. Rose, and V. Z. G., “A detailed router for Field-Programmable Gate Arrays,” *IEEE Transactions on CAD*, pp. 620–628, May 1992.
- [18] M. L. Bushnell and V. D. Agrawal, “Digital DFT and scan design,” in *Essential of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. New York, NY: Springer, 2000, ch. 14.

- [19] *ASSURA parasitic extraction data sheet (5772 10/04)*, Cadence Design Systems, Inc, 2004.
- [20] *Nano Encounter data sheet (4137H 08/04)*, Cadence Design Systems, Inc, 2004.
- [21] J. Cong and Z. Pan, “Interconnect performance estimations models for design planning,” *IEEE Transactions on Computer-Aided Design*, vol. 20, pp. 739–752, June 2001.
- [22] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press/McGraw-Hill, 1990.
- [23] F. Corno, M. Sonza Reorda, and G. Squillero, “RT-level ITC’99 benchmarks and first ATPG results,” *IEEE Design & Test of Computers*, pp. 44–53, July-August 2000.
- [24] D. C. Cronquist and L. MacMurchie, “Emerald- an architecture-driven tool compiler for FPGAs,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, 1996, pp. 144–150.
- [25] K. K. David Chinnery, *Closing the Gap between ASIC and Custom: Tools and Techniques for High-Performance ASIC Design*. Kluwer Academic Publishers (now Springer), 2002.
- [26] J. A. Davis and J. D. Meindl, Eds., *Interconnect Technology and design for Gigascale Integration*. New York, NY: Springer, 2003.
- [27] V. Degalahal and T. Tuan, “Methodology for high level estimation of FPGA power consumption,” in *Proc. of 10th Asia and South Pacific Design Automation Conference (ASP-DAC 2005)*, Shanghai, China, 2005.

- [28] A. DeHon, “Balancing interconnect and computation in a reconfigurable computing array (or, why you don’t really want 100% LUT utilization),” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, 1999, pp. 21–23.
- [29] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
- [30] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns, “Placement and routing tools for the Triptych FPGA,” *IEEE Transactions on VLSI*, pp. 473–482, December 1995.
- [31] W. C. Elmore, “The transient response of damped linear networks with particular regard to wideband amplifiers,” *Journal of Applied Physics*, pp. 55–63, January 1948.
- [32] *Structured ASIC technology (FTC-EE31)*, Faraday Technology Corp., 2005.
- [33] (2003) Fujitsu AccelArray. [Online]. Available: <http://www.fujitsu.com/emea/services/microelectronics/asic/accelarray/>
- [34] J. M. Green and H. Klar, “CMOS gate array architecture for digital signal processing applications,” *IEEE Journal of Solid State Circuits*, vol. 31, no. 3, pp. 410–418, March 1996.
- [35] J. N. Helbert, *Handbook of VLSI Microlithography- Principles, Tools, Technology and Applications, Second Edition*. William Andrew Publishing/ Noyes, 2001.
- [36] B. Hu, H. Jiang, Q. Liu, and M. Marek-Sadowska, “Synthesis and placement flow for gain-based programmable regular fabrics,” in *Proceedings of International Symposium on Physical Design(ISPD’03)*, Monterey, CA, 2003, pp. 197–203.

- [37] *C11N Design Manual, version 3.0*, Infineon Technologies, 2003.
- [38] *Memory Explorer tool (version: R2.4.2)*, Infineon Technologies AG, 2005.
- [39] Y. Ismail, E. Friedman, and J. Neves, “Figures of merit to characterize the importance of on-chip inductance,” in *Proceedings of the Design Automation Conference*, 1998, pp. 560–565.
- [40] X. Jiang, “Power consumption studies in FPGAs,” Master’s thesis, University of Aachen, Aachen, Germany, Feb. 2006.
- [41] S. P. Khatri, A. Mehrotra, R. K. Bryton, R. H. J. M. Otten, and A. Sangiovanni-Vincentelli, “A novel VLSI layout fabric for deep sub-micron application,” *Proceedings of the 36th ACM/IEEE Design Automation Conference*, pp. 491–496, June 1999.
- [42] S. Koeppe, Infineon Technologies AG, private communication, 2006.
- [43] A. Koorapaty, P. Kheterpal, P. Gopalakrishnan, M. Fu, and L. Pileggi, “Exploring logic block granularity for regular fabrics,” in *Proceedings of Design, Automation and Test in Europe Conference and Exposition (DATE 2004)*, Paris, France, 2004, pp. 468–473.
- [44] C. Y. Lee, “An algorithm for path connections and its applications,” *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, Sept. 1961.
- [45] G. Lemieux and D. Lewis, *Design of Interconnection Networks for Programmable Logic*. Norwell, MA: Kluwer Academic Publishers, 2004.
- [46] A. Levinthal and R. Herveille, “FlexASIC structured array: A solution to the DSM challenge,” *DesignCon 2005*, February 2005.

- [47] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, “Power modeling and characteristics of field programmable gate arrays,” *IEEE Transactions on Computed-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 1712–1724, 2005.
- [48] (2006) The lightspeed website. [Online]. Available: <http://www.lightspeed.com/>
- [49] J. Lin, A. Jagannathan, and J. Cong, “Placement-driven technology mapping for LUT-based FPGAs,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, 2003, pp. 121–126.
- [50] *ModelSim SE Data Sheet (1023780-w)*, Mentor Graphics Corp., 2005.
- [51] T. Miyasaki, A. Takahara, T. Murooka, M. Katayama, T. Ichimori, K. Shirakawa, A. Tsutsui, and K. Fukami, “PROTEUS-Lite project: Dedicated to developing a telecommunication-oriented FPGA and its applications,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 4, pp. 401–414, August 2000.
- [52] J. A. Nestor, “A new look at hardware maze routing,” in *ACM Great Lakes Symposium on VLSI*, 2002, pp. 142–147.
- [53] M. Okabe, Y. Okuno, T. Arakawa, I. Tomioka, T. Ohno, T. Noda, M. Hatanaka, and Y. Kuramitsu, “A 400k-transistor CMOS sea-of-gate array with continuous track allocation,” *IEEE Journal of Solid State Circuits*, vol. 24, pp. 1280–1286, October 1989.
- [54] T. Okamoto, T. Kimoto, and N. Maeda, “Design methodology tools for NEC electronics’ structured ASIC ISSP,” *Proceedings of the ISPD’04*, pp. 90–96, April 2004.

- [55] (2006) Opencores.org: free open source IP cores and chip design. [Online]. Available: <http://www.opencores.org/>
- [56] R. Padovani, “HardWire LCAs: A risk-free FPGA-to-ASIC migration path,” in *Proceedings of the IEEE WESCON’95 Conference*, San Francisco, CA, 1995, pp. 272–277.
- [57] C. Patel, A. Cozzie, H. Schmit, and L. Pileggi, “An architectural exploration of via patterned gate arrays,” in *Proceedings of International Symposium on Physiscal Design(ISPD’03)*, Monterey, CA, 2003, pp. 184–189.
- [58] H. Patel, “Accelerate design performance using Xplorer,” *Xilinx Xcell Journal*, vol. 55, 2005.
- [59] —, “Better...stronger...faster,” *Xilinx Xcell Journal*, vol. 49, 2004.
- [60] S. Perry, G. Nixon, L. Kong, A. Scott, A. Hall, L. Wang, C. Dettmar, J. Park, and R. Price, “Method for programming a mask-programmable logic device and device so programmed,” U.S. Patent 2004/0 261 052 A1, 23, 2004.
- [61] L. T. Pileggi, H. Schmit, A. J. Strojwas, P. Gopalakrishnan, V. Kheterpal, A. Koorapaty, C. Patel, V. Rovner, and K. Y. Tong, “Exploring regular fabrics to optimize the performance-cost trade-off,” in *DAC*, 2003, pp. 782–787.
- [62] J. Rabaey, A. Chandrakasan, and B. Nikolic, “Designing complex digital integrated circuits,” in *Digital Integrated Circuits: a design perspective, 2nd edition*. New Jersey, PH: Prentice-Hall International, 2003, ch. 8. [Online]. Available: <http://bwrc.eecs.berkeley.edu/IcBook/slides.htm>
- [63] M. A. Rahman, MorethanIP GmbH, private communication, 2005.

- [64] ———, “Layout studies of a Mask-Programmable substitute of an FPGA Tile,” Master’s thesis, University of Ulm, Ulm, Germany, Oct. 2004.
- [65] Y. Ran and M. Marek-Sadowska, “Designing via-configurable logic block for regular fabric,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 1–14, 2006.
- [66] J. Rubenstein, P. Penfield, and A. Horowitz, “Signal delay in RC tree networks,” *IEEE Transactions on Computer-Aided Design*, vol. CAD-2, pp. 202–211, 1983.
- [67] T. Sakurai, “Perspectives of low-power VLSI’s,” *IEICE Transactions on Electronics*, vol. E87-C, pp. 429–436, 2004.
- [68] M. Santarini, “Structured ASIC deserve serious attention at 90nm,” *EDN Magazine*, pp. 69–74, July 2005.
- [69] S. Sapatnekar, *Timing*. Norwell, MA: Kluwer Academic Publishers, 2004.
- [70] M. Scheppler, “Area reduction potential of a Hardened FPGA,” Infineon Technologies AG, Germany, Tech. Rep. Internal Report, Jan. 2006.
- [71] *PowerTheater Data Sheet*, Sequence Design, Inc., 2005.
- [72] L. Shang, A. Kaviani, and K. Bathala, “Dynamic power consumption in Virtex-II FPGA family,” in *Proceedings of the ACM/SIGDA 10th International Symposium on Field Programmable Gate Arrays*, Monterey, CA, 2002, pp. 157–164.
- [73] K. L. Shepard, “Practical issues of interconnect analysis in deep submicron integrated circuits,” in *Proceedings of the International Conference on Computer Design*, San Jose, CA, 1997, pp. 532–541.

- [74] D. D. Sherlekar, O. Siguenza, and H. Yang, “Maximize design flexibility with fast turnaround time while minimizing design cost with metal programmable libraries,” *IP Based Design 2003 Conference*, November 2003.
- [75] N. A. Sherwani, *Algorithms for VLSI Physical Automation, Third Edition*. Norwell, MA: Kluwer Academic Publishers, 1999.
- [76] M. J. S. Smith, “Circuit extraction and DRC,” in *Application-Specific Integrated Circuits*. Boston, MA: Addison-Wesley Publishing Company, 1997, ch. 17.4.
- [77] D. Sylvester and K. Keutzer, “A global wiring paradigm for deep sub-micron design,” *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 242–252, February 2000.
- [78] “PrimeTime: Synopsys static timing solution,” White Paper, Synopsys, Inc, Synopsys, Inc, 1999.
- [79] *PrimeTime User guide: Fundamentals (X-2005.12)*, Synopsys, Inc, 2005.
- [80] *NanoSim Memory and Mixed-signal Verification (10/03.MH.WO 03-11743)*, Synospys, Inc, 2003.
- [81] *Design Compiler 2005 data sheet (05/05.KF.WEB.05-132787)*, Synospys, Inc, 2005.
- [82] *Synplify Pro data sheet (2090SSP)*, Synplicity, Inc, 2005.
- [83] T. Taghavai, S. Ghiasi, A. Ranjan, S. Raje, and M. Sarrafzadeh, “Innovate or perish: FPGA physical design,” in *Proc. International Symposium on Physiscal Design(ISPD’04)*, Phoenix, USA, 2004, pp. 148–155.

- [84] D. Tavana, W. Yee, S. Young, and B. Fawcell, “Logic block and routing considerations for a new SRAM-based FPGA architecture,” *Proceedings of the IEEE 1995 Custom Integrated Conference*, pp. 511–514, April 1995.
- [85] M. Tom and G. Lemieux, “Logic block clustering of large designs for channel-width constrained FPGAs,” in *Proc. Design Automation Conference (DAC’05)*, Anaheim, USA, 2005.
- [86] K. Y. Tong, V. Kheterpal, V. Rovner, L. Pileggi, H. Schmit, and R. Puri, “Regular logic fabrics for a via patterned gate array (VPGA),” in *Proceedings of IEEE Custom Integrated Circuits Conference*, San Jose, CA, 2003, pp. 53–56.
- [87] L. P. P. P. van Ginneken, “Buffer placement in distributed RC-tree networks for minimal Elmore delay,” *International Symposium on Circuits and Systems*, pp. 865–868, May 1990.
- [88] H. Veendrick, *Deep-Submicron CMOS ICs*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000.
- [89] F. J. Veredas, “Associative search processor design in VHDL,” Master’s thesis, Autonomic University of Barcelona, Bellaterra, Spain, 2003.
- [90] F. J. Veredas and H. J. Pfleiderer, “Automated conversion from a LUT-based FPGA to a LUT-based MPGA,” in *Proceedings of the International Conference on Field Programmable Logic and its Applications (FPL’06)*, Madrid, Spain, 2006.
- [91] —, “Power estimation of a LUT-based MPGA,” in *Proceedings of the IEEE International Conference on Field Programmable Technology (FPT’06)*, Bangkok, Thailand, 2006.

- [92] F. J. Veredas, M. Scheppler, C. Heer, J. Carrabina, and H. J. Pfeiderer, “FPGA accelerator for an associative computing engine,” in *Proc. of the III Workshop on Reconfigurable Computing and Applications*, Madrid, Spain, 2003, pp. 433–440.
- [93] F. J. Veredas, M. Scheppler, and H. J. Pfeiderer, “A survey on reconfigurable computing systems: taxonomy and metrics,” in *Proc. of the IV Workshop on Reconfigurable Computing and Applications*, Barcelona, Spain, 2004, pp. 24–36.
- [94] —, “Automated conversion from a LUT-based FPGA to a LUT-based MPGA with fast turnaround time,” in *Proc. Design Automation Europe (DATE’06)*, Munich, Germany, 2006.
- [95] F. J. Veredas, M. Scheppler, B. Zhai, and H. J. Pfeiderer, “LUT-based MPGAs for fast turnaround time conversion flow,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS’06)*, Kos, Greece, May 2006.
- [96] —, “Regular routing architecture for LUT-based MPGA,” in *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI’06)*, Karlsruhe, Germany, 2006.
- [97] *ViaMask: One-mask structured ASIC library (2004.2.04)*, ViASIC, Inc., 2004.
- [98] S. Wenande and R. Chidester, “Power analysis to new levels with XPower,” *Xilinx Xcell Journal*, pp. 26–27, Jan. 2001.
- [99] *Virtex-II ProTM Platform FPGA Handbook (UG012,v2.0)*, Xilinx Inc., 2002.
- [100] *Xilinx Synthesis Technology (XST) user guide*, Xilinx Inc., 2002.

- [101] *ISE 8.1i Data Sheet (PN0010867-03)*, Xilinx Inc., 2006.
- [102] (2006) The Xilinx website. [Online]. Available: <http://www.xilinx.com/>
- [103] S. P. Young, “Expandable interconnect structure for FPGAs,” U.S. Patent 6 396 303 B1, May 28, 2002.
- [104] B. Zhai, “Physical implementation studies of an MPGA,” Master’s thesis, University of Ulm, Ulm, Germany, 2006.
- [105] T. Zhang and S. Sapatnekar, “Buffering global interconnects in structured ASIC design,” *Proceedings of the Asia/South Pacific Design Automation Conference*, January 2005.
- [106] Q. K. Zhu, “IR voltage drop,” in *Power Distribution Network Design for VLSI*. Hoboken, NJ: John Wiley & Sons, Inc., 2004, ch. 4.
- [107] ———, “Microprocessor clock distribution examples,” in *High-Speed Clock Network Design*. Boston, NY: Kluwer Academic Publishers, 2003, ch. 6.
- [108] P. S. Zuchowski, C. B. Reynolds, R. J. Grupp, S. G. Davis, B. Cremen, and B. Troxel, “A hybrid ASIC and FPGA architecture,” in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design (ICCAD ’02)*, San Jose, CA, 2002, pp. 187–194.