

**METHODOLOGY AND ECOSYSTEM**  
**FOR THE DESIGN OF A**  
**COMPLEX NETWORK ASIC**



Inauguraldissertation  
zur Erlangung des akademischen Grades  
eines Doktors der Naturwissenschaften  
der Universität Mannheim

vorgelegt von

Sven Uwe Kapferer

(Diplom-Informatiker der Technischen Informatik)

aus Mannheim

Mannheim, 2012

**Dekan:** Prof. Dr. Heinz Jürgen Müller, Universität Mannheim  
**Referent:** Prof. Dr. Ulrich Brüning, Universität Heidelberg  
**Korreferent:** Prof. Dr. Holger Fröning, Universität Heidelberg

Tag der mündlichen Prüfung: 14.02.2013

# Abstract

Performance of HPC systems has risen steadily. While the 10 Petaflop/s barrier has been breached in the year 2011 the next large step into the exascale era is expected sometime between the years 2018 and 2020. The EXTOLL project will be an integral part in this venture. Originally designed as a research project on FPGA basis it will make the transition to an ASIC to improve its already excelling performance even further. This transition poses many challenges that will be presented in this thesis. Nowadays, it is not enough to look only at single components in a system. EXTOLL is part of complex ecosystem which must be optimized overall since everything is tightly interwoven and disregarding some aspects can cause the whole system either to work with limited performance or even to fail.

This thesis examines four different aspects in the design hierarchy and proposes efficient solutions or improvements for each of them. At first it takes a look at the design implementation and the differences between FPGA and ASIC design. It introduces a methodology to equip all on-chip memory with ECC logic automatically without the user's input and in a transparent way so that the underlying code that uses the memory does not have to be changed. In the next step the floorplanning process is analyzed and an iterative solution is worked out based on physical and logical constraints of the EXTOLL design. Besides, a work flow for collaborative design is presented that allows multiple users to work on the design concurrently. The third part concentrates on the high-speed signal path from the chip to the connector and how it is affected by technological limitations. All constraints are analyzed and a package layout for the EXTOLL chip is proposed that is seen as the optimal solution. The last part develops a cost model for wafer and package level test and raises technological concerns that will affect the testing methodology. In order to run testing internally it proposes the development of a stand-alone test platform that is able to test packaged EXTOLL chips in every aspect.



# Zusammenfassung

Die Leistung von HPC Systemen hat sich kontinuierlich gesteigert. Im Jahr 2011 wurde die 10 Petaflop/s Grenze durchbrochen. Der nächste große Schritt in das Exascale Zeitalter wird nun für irgendwann zwischen den Jahren 2018 und 2020 erwartet. Das EXTOLL Projekt wird ein wesentlicher Bestandteil auf dem Weg dorthin sein. Ursprünglich für FPGA Technologie entwickelt wird im nächsten Schritt die Umsetzung als ASIC verwirklicht, um die schon jetzt ausgezeichnete Leistung noch weiter zu steigern. Dieser Wechsel birgt viele Herausforderungen, die in dieser Arbeit vorgestellt werden. Heutzutage genügt es nicht, nur die Einzelkomponenten eines Systems zu betrachten. EXTOLL ist ein Teil eines komplexen Ganzen und muss an allen Stellen optimiert werden, da alle Einzelteile eng miteinander verbunden sind. Die Vernachlässigung einzelner Aspekte kann dazu führen, dass das Gesamtsystem entweder nur mit begrenzter Leistung oder vielleicht sogar gar nicht funktioniert.

Diese Arbeit untersucht vier unterschiedliche Aspekte im Designablauf und stellt für jeden dieser Aspekte eine effiziente Lösung oder Verbesserung vor. Zuerst wird die Umsetzung des Designs betrachtet und die Unterschiede zwischen einem FPGA und einem ASIC Design. Es wird eine Methodik vorgestellt, um den Speicher auf dem Chip automatisch und ohne Zutun des Benutzers mit ECC Logik auszustatten und zwar so, dass der Code, der den Speicher benutzt, nicht geändert werden muss. Im nächsten Schritt wird der Ablauf des Floorplanning analysiert und eine iterative Lösung des Problems basierend auf den technischen und internen Randbedingungen herausgearbeitet. Außerdem wird ein Arbeitsablauf für gemeinschaftliches Arbeiten vorgestellt, der es mehreren Benutzern erlaubt, gleichzeitig am Design zu arbeiten. Der dritte Teil konzentriert sich auf den Hochgeschwindigkeitspfad vom Chip bis zum Stecker und welchen Einfluss die technologischen Einschränkungen auf ihn haben. Alle Bedingungen werden analysiert und ein Entwurf eines Package für den EXTOLL Chip wird aufgezeigt, der als bestmögliche Lösung angesehen wird. Im letzten Teil wird ein Kostenmodell für Tests auf dem Wafer und im Package entwickelt. Es werden technologische Bedenken geäußert, die Einfluss auf den Testablauf haben. Um Tests im Labor selbst durchführen zu können, wird die Entwicklung einer selbständigen Testplattform vorgeschlagen, die es erlaubt fertige EXTOLL Chips nach allen Gesichtspunkten zu testen.



# Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>1</b>
1.1	Motivation.....	1
1.2	Scope of Work.....	3
1.3	EXTOLL Design .....	4
1.3.1	Host Interface .....	5
1.3.2	EXTOLL Core Logic.....	6
1.3.3	EXTOLL NIC.....	8
1.4	Outline .....	8
<b>2</b>	<b>EXTOLL ASIC.....</b>	<b>11</b>
2.1	Design Decisions .....	11
2.1.1	Performance Comparison.....	11
2.1.2	Technology Analysis.....	12
2.2	Transition from FPGA to ASIC.....	17
2.3	Design Guidelines.....	17
2.3.1	Timing Basics.....	18
2.3.2	Reset.....	19
2.4	Embedded Memory.....	21
2.4.1	Memory Faults.....	23
2.4.2	Automatic Memory Generator.....	26
2.5	Design for Test.....	31
2.6	Clocking.....	32
2.6.1	Supporting Circuits.....	33
2.6.2	Clocking Structure .....	35
2.6.3	PLL.....	37

2.7	Prototyping.....	38
2.8	Toplevel Organization.....	41
<b>3</b>	<b>ASIC Design Considerations.....</b>	<b>47</b>
3.1	Design Preparation .....	47
3.1.1	LIB.....	47
3.1.2	LEF.....	48
3.1.3	Capacitance Tables.....	49
3.1.4	SDC / Timing Constraints.....	50
3.1.5	DEF.....	51
3.1.6	Flow Script.....	51
3.2	Data Hierarchy .....	52
3.3	Frontend Flow .....	54
3.4	Backend Flow.....	60
3.5	Floorplanning and Datapath Analysis .....	63
3.5.1	Datapath Analysis - Global View.....	66
3.5.2	Pre-placement .....	67
3.5.3	Datapath Analysis - Detailed View.....	69
3.5.4	Miniature Optimizations .....	74
<b>4</b>	<b>Optimization of Complex Interconnection Structures.....</b>	<b>79</b>
4.1	Design constraints.....	79
4.1.1	Technological Limitations .....	80
4.1.2	Signal Integrity .....	80
4.1.3	Viability.....	80
4.1.4	Economic Feasibility .....	80
4.2	Design Components .....	81
4.2.1	Connector .....	81

4.2.2	PCB .....	83
4.2.3	Package .....	86
4.2.4	Die .....	90
4.3	Automatic Generation .....	91
4.4	SI Analysis .....	92
4.5	PDN Design .....	93
4.6	Results .....	95
4.6.1	EXTOLL I/O .....	95
4.6.2	EXTOLL Supply .....	101
4.6.3	Constraints and Efficiency .....	103
<b>5</b>	<b>EXTOLL Test .....</b>	<b>105</b>
5.1	Test Analysis .....	105
5.2	Wafer Test .....	107
5.3	Package Test .....	108
5.4	Process Analysis .....	109
5.5	Test Setup .....	113
5.6	Test Hardware .....	116
5.6.1	Analysis .....	116
5.6.2	Proposal .....	120
<b>6</b>	<b>Conclusion .....</b>	<b>125</b>
6.1	Results .....	125
6.2	Project Review .....	127
6.3	Outlook .....	128
<b>A</b>	<b>Acronyms .....</b>	<b>131</b>

<b>B</b>	<b>Bibliography</b> .....	<b>137</b>
<b>C</b>	<b>EXTOLL Package</b> .....	<b>147</b>
<b>D</b>	<b>STIL Example</b> .....	<b>153</b>

# List of Figures

Figure 1.1 TOP500 Development .....	1
Figure 1.2 DEEP Architecture .....	2
Figure 1.3 Booster Node Architecture .....	3
Figure 1.4 EXTOLL Block Diagram .....	5
Figure 2.1 Design Space Diagram for EXTOLL Technology Selection .....	13
Figure 2.2 Flip-Flop Timing Parameters.....	18
Figure 2.3 Reset Synchronizer Circuit.....	20
Figure 2.4 FPGA Memory Flow.....	21
Figure 2.5 Memory Size Distribution.....	22
Figure 2.6 Memory Speed Analysis .....	23
Figure 2.7 Occupied Chip Area.....	24
Figure 2.8 ECC Complexity Analysis.....	26
Figure 2.9 256x128 RAM Instance.....	27
Figure 2.10 RAM Timing with Handshaking.....	29
Figure 2.11 Levels of Indirection .....	29
Figure 2.12 Debug Port Overview.....	32
Figure 2.13 Divide-by-3 Clock Divider.....	33
Figure 2.14 Divide-by-3 Clock Divider Waveform .....	33
Figure 2.15 Divide-by-5 Clock Divider.....	34
Figure 2.16 Divide-by-5 Clock Divider Waveform .....	34
Figure 2.17 Glitchless Clock Multiplexer Waveform.....	34
Figure 2.18 Glitchless Clock Multiplexer.....	35
Figure 2.19 EXTOLL Clocking Scheme .....	36
Figure 2.20 EXTOLL PLL .....	37

Figure 2.21 Ventoux Prototyping Platform .....	39
Figure 2.22 Galibier Prototyping Platform .....	40
Figure 2.23 PCIe Backplane .....	40
Figure 2.24 FPGA Toplevel Organization .....	41
Figure 2.25 ASIC Toplevel Organization .....	43
Figure 3.1 Layer Stack of EXTOLL ASIC .....	48
Figure 3.2 Effect of Optical Proximity Correction .....	48
Figure 3.3 Simple Inverter as Full Layout and Abstract View .....	49
Figure 3.4 EXTOLL ASIC Subversion Structure .....	53
Figure 3.5 Gate vs. Interconnect Delay .....	55
Figure 3.6 Synthesis Input and Output Files .....	55
Figure 3.7 Frontend Flow Organization .....	57
Figure 3.8 Memory Test Connections .....	59
Figure 3.9 Backend Flow Organization .....	61
Figure 3.10 Signal Reach in One Clock Cycle (8 FO4) .....	64
Figure 3.11 Automatic Placement without Floorplan .....	65
Figure 3.12 Global Datapath Analysis .....	67
Figure 3.13 Automatic Placement with Preplaced Macros .....	68
Figure 3.14 Hierarchical Floorplanning .....	69
Figure 3.15 HTAX Structure .....	72
Figure 3.16 EXTOLL XBAR Structure .....	73
Figure 3.17 Relative Placement Example .....	74
Figure 3.18 Schematic of a 4x2 Register Based RAM .....	76
Figure 4.1 Connectivity Design Constraints .....	79
Figure 4.2 Schematic Representation of Signal Path .....	81
Figure 4.3 Samtec HDI6 Connector .....	82
Figure 4.4 PCB Layout of HDI6 Connector .....	82
Figure 4.5 Impact of Parameters on Impedance .....	84

Figure 4.6 Differential Stripline .....	84
Figure 4.7 Example PCB Stackup.....	85
Figure 4.8 Aspin Test Board .....	86
Figure 4.9 Differential Pair BGA Breakout.....	88
Figure 4.10 A 12x Serdes Breakout.....	89
Figure 4.11 Scripting Approach.....	91
Figure 4.12 Single Channel Simulation Setup.....	92
Figure 4.13 Eye Diagram at 10 Gb/s.....	93
Figure 4.14 Example of an Insufficient Power Supply .....	94
Figure 4.15 Oscillator Circuit.....	98
Figure 5.1 Test Options.....	105
Figure 5.2 Chip Test Flow.....	106
Figure 5.3 Teradyne Tester .....	107
Figure 5.4 Microprobe Vx-MP Probe Card.....	107
Figure 5.5 Spring Pin .....	108
Figure 5.6 Test Cost Analysis .....	112
Figure 5.7 Scan Chain Structure .....	114
Figure 5.8 EXTOLL Test Organization.....	115
Figure 5.9 CMOS Inverter .....	118
Figure 5.10 Test Data Processing.....	120
Figure 5.11 Design Space for Tester Development.....	120
Figure 5.12 Test Platform Proposal .....	122



# List of Tables

Table 2.1 Competitor Overview.....	11
Table 2.2 Timing Parameters and Their Meaning .....	18
Table 3.1 Design Data Storage.....	54
Table 3.2 Floorplan Evaluation .....	70
Table 3.3 XBAR Area Distribution.....	73
Table 4.1 EXTOLL High Speed Connectivity .....	79
Table 4.2 Connector Overview.....	81
Table 4.3 HyperTransport Interface Pins .....	96
Table 4.4 PCI Express Interface Pins.....	96
Table 4.5 Network Interface Pins.....	97
Table 4.6 Clocking Pins.....	97
Table 4.7 Configuration and Reset Pins.....	98
Table 4.8 Miscellaneous Pins.....	99
Table 4.9 JTAG Interface Pins.....	100
Table 4.10 Dedicated DFT Pins .....	101
Table 4.11 Digital Core and I/O Supply.....	102
Table 4.12 HyperTransport Supply .....	102
Table 4.13 Clocking Supply .....	103
Table 4.14 Serializer Supply.....	103
Table 5.1 Parameters for Die Calculation.....	109
Table 5.2 Parameters for Cost Calculation.....	111



# 1 Introduction

## 1.1 Motivation

*“I think there is a world market for maybe five computers”*

Alleged quote by Thomas Watson, IBM CEO, 1943

Regardless of the historical accuracy of the quote it represents a view that no one could imagine that there is a need for actual computing power. Nevertheless, performance of processors has risen steadily and continues growing exponentially. On the basis of Moore’s law that states that the number of transistors in an integrated circuit doubles every two years it was predicted that the performance doubles every 18 months. This prediction has even been surpassed by the supercomputer market.

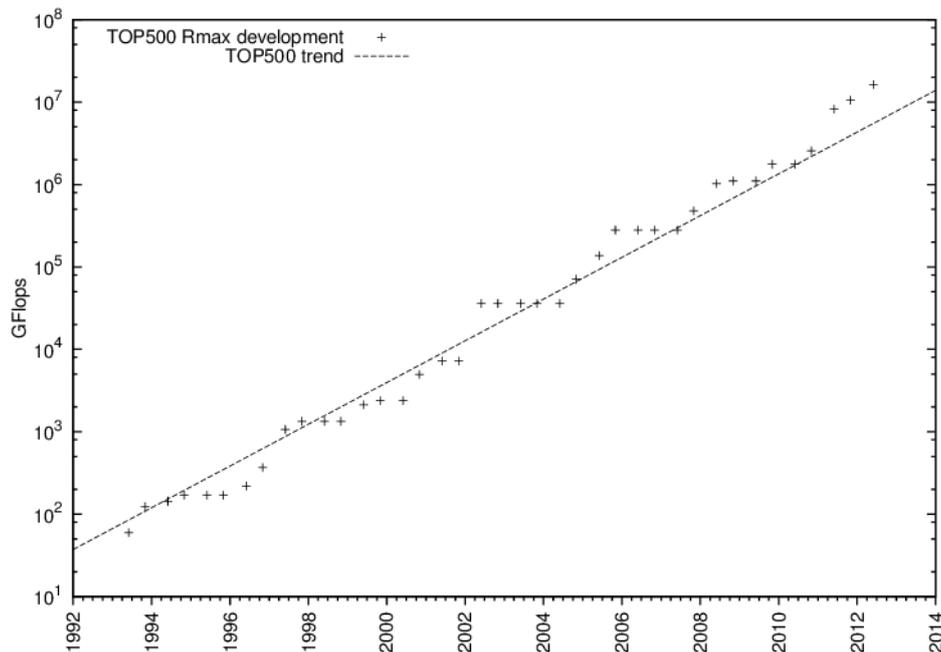


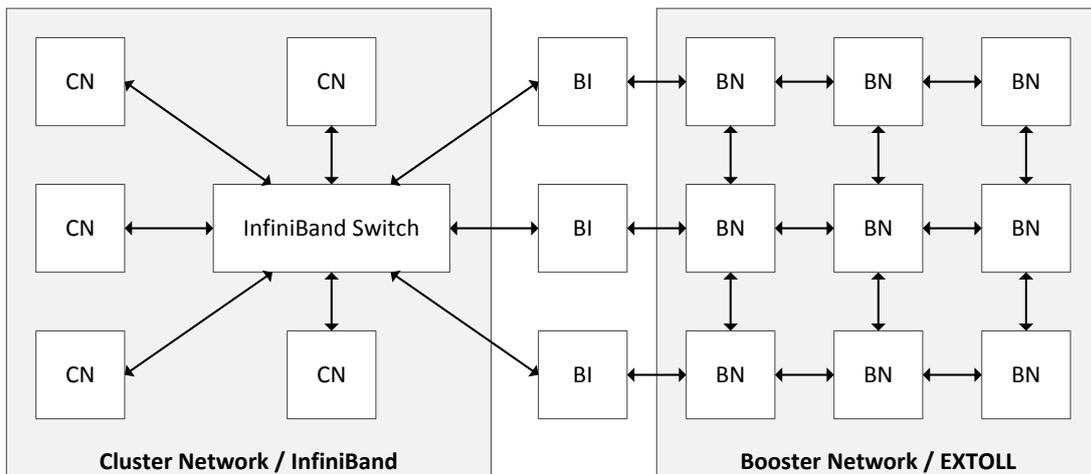
Figure 1.1 TOP500 Development

## Introduction

---

A historical observation from the Top500 list<sup>1</sup> [1] in figure 1.1 shows that the maximum performance of the fastest installation which is measured by the Linpack benchmark [2] has increased tenfold about every four years. With the first installation breaching the 10 Petaflop/s barrier in November 2011 it can be projected that the Exascale era will be reached around 2020. However, current architectures will not scale up to this point [3]. One project to meet the Exascale challenge is the DEEP (Dynamic Exascale Entry Platform) project which is supported by the European Union through the Seventh Framework Programme (FP7).

The idea is to combine two separate networks. One is a typical cluster installation (CN) connected through a central InfiniBand switch. The other network is a specialized booster network consisting of accelerators (BN) that are connected in a torus topology through a specialized high-performance, low latency network, EXTOLL. The following figure 1.2 shows the organization of the DEEP architecture and how the two networks are connected through so called Booster Interconnect (BI) nodes.

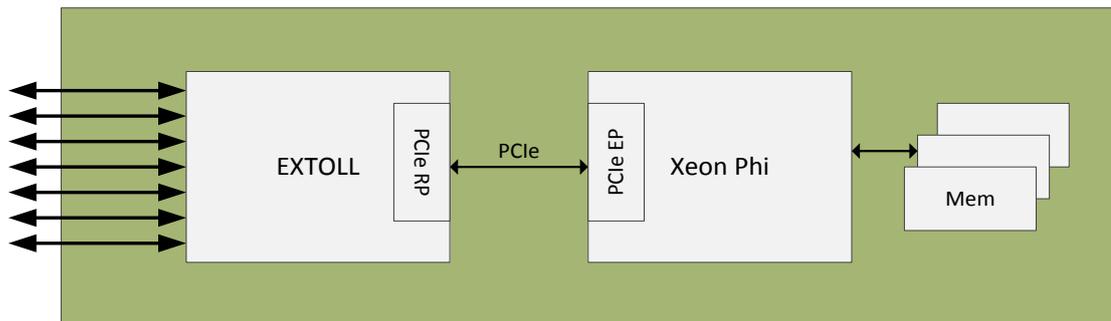


**Figure 1.2 DEEP Architecture**

The following figure 1.3 shows a schematic representation of a Booster Node. At the moment it is planned that one hardware node will consist of two booster nodes.

---

<sup>1</sup> Based on the 39<sup>th</sup> list from June 2012



**Figure 1.3 Booster Node Architecture**

The noteworthy feature is that the system does not need a host CPU, the accelerator (Xeon Phi) is directly connected to the network through EXTOLL.

## 1.2 Scope of Work

The International Technology Roadmap for Semiconductors (ITRS) identified “Networking” in their 2011 report [4] as one of the key markets that drives the future of the semiconductor industry. The need for higher bandwidth which is estimated to quadruple every 3-4 years, the short time-to-market, high reliability and the need to keep the power envelope low lead to advances in serializer technology, embedding of switching functionality and the ability to integrate a larger number of gates in a chip to move all system functions into a single large System-on-Chip (SoC). An analysis of the 10 fastest computers in the Top500 list suggests that a custom / proprietary interconnect that is tailored specifically for the system is still the best path to take. Such a custom interconnect that was selected for the DEEP system is the EXTOLL interconnect which was specifically designed for the High Performance Computing (HPC) market. Because it is an integral part of the functionality of the Booster Network as seen in figures 1.2 and 1.3 it is especially important that it will have the highest performance available so that the whole system will perform outstandingly. Highest performance means that EXTOLL must be implemented as an ASIC [5] which allows both a high internal frequency and also fast connectivity to the outside world. This cannot be achieved with an FPGA since the performance gap between FPGAs and ASICs is rather large despite continuous improvements in FPGA technology. For a purely logic based design an ASIC will perform about 4 times faster than an FPGA [6].

However, a system does not only consist of a single component but is a complex ecosystem that represents a pretty wide design space. With an ASIC that is designed from scratch this

ecosystem can be optimized for all components. But this multi-dimensional optimization problem must be solved concurrently since each decision has a direct impact on both the performance and the optimization space of the remaining components.

This thesis analyzes four different aspects of the EXTOLL chip and its surrounding ecosystem and either proposes solutions or methodologies that significantly improve the final results.

- **Design**

The key differences between FPGA and ASIC design are worked and out, especially in the area of on-chip memory and an innovative methodology is introduced to handle on-chip error checking and correction in a transparent way.

- **Physical implementation**

Special consideration is given to the floorplanning process and how it relates to both physical constraints like signal arrangement and logical constraints like the dataflow inside the design.

- **Connectivity and signal integrity**

The signal path from the chip, through the package and the PCB to the connector is analyzed and a package definition is proposed that is an optimal solution considering the technological limitations, viability, signal integrity and also cost concerns.

- **Testing**

At last a cost model concerning test is developed and as a consequence of this a new test platform is presented that is able to test packaged EXTOLL chips in every aspect.

### 1.3 EXTOLL Design

EXTOLL (EXtended aTOLL) is the successor of ATOLL, the first network chip developed at the Chair of Computer Architecture, University of Mannheim, which was implemented in a 180nm process from UMC [7]. The good performance numbers and many promising ideas lead to the next research project EXTOLL which has been in development for the last years.

EXTOLL can be logically divided into three different parts:

- The host interface part, which contains two exclusive commodity host interface controllers

- The EXTOLL core logic part, which contains both the FUs (functional units) that make up EXTOLL's basic functionality and assisting units that are needed to support all features of EXTOLL
- The network interface part which includes the switching functionality as well as the control logic to interface to the outside world

The block diagram in figure 1.4 gives an overview of the design blocks inside EXTOLL that will be shortly described in the following.

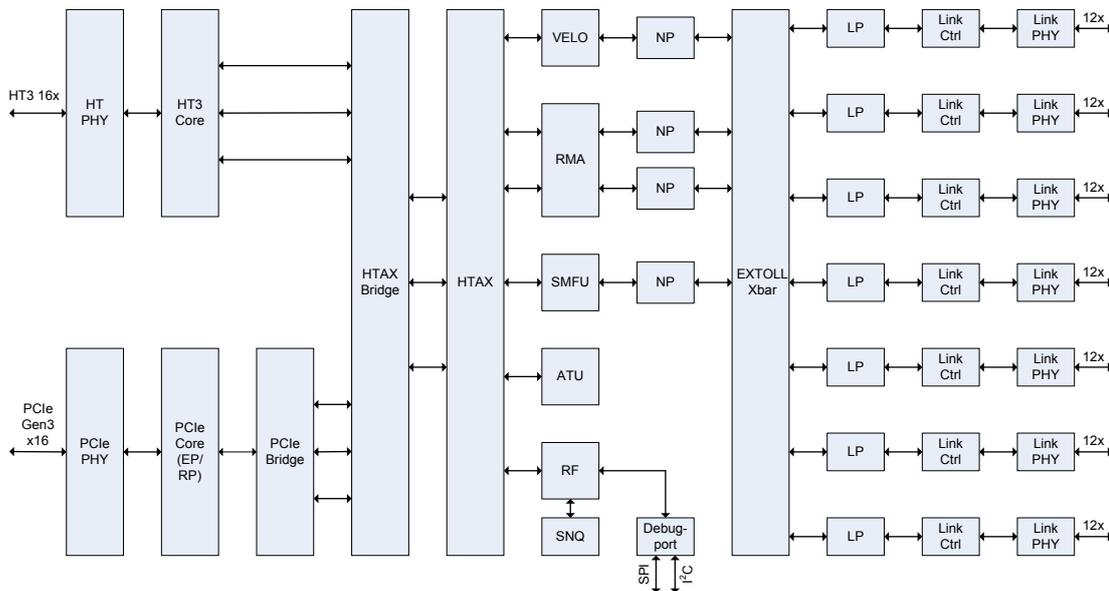


Figure 1.4 EXTOLL Block Diagram

### 1.3.1 Host Interface

One of the supported host interfaces is HyperTransport (HT) [8] that is specific to AMD based systems. The *HT3 core* [9] inside of EXTOLL is a standard compliant implementation of the protocol supporting all required features to directly connect to modern AMD processors. The HT3 core is connected to the *HT PHY* that takes care of the physical layer implementation of the HT links similar to the PHY implementations in the CPU [10] [11]. With a link width of 16 bit and supported transfer rates of up to 2.6 GT/s the link is able to sustain a raw bandwidth of 10.8 GB/s.

Besides HyperTransport EXTOLL also supports PCI Express (PCIe) [12] as a host interface. Both the *PCIe PHY* that handles the physical layer and the *PCIe core* that implements the

data link layer and transaction layer are external IP and are not developed in-house. They both support the latest 3.0 revision of the specification, thus providing up to 16 GB/s link bandwidth for an x16 connection to the host system. The PCIe core can be configured either as an endpoint, thereby assuming the role of a traditional network interface controller (NIC), and also as a rootport. Working as a rootport allows a direct host-independent communication with an endpoint, e.g. a graphics card. Therefore an arbitrary device can be added to the network without the need for a host CPU. Although HyperTransport and PCIe share architectural similarities they are not directly interchangeable. Thus, a translation layer, the *PCIe Bridge* [13], is needed to generate packets that can be understood by the following modules and to handle the peculiarities of the PCIe core and protocol.

The *HTAX Bridge* [14] is the interface between the host interfaces and the internal EXTOLL logic. In RX direction it translates the packet format of the IP cores to the HTOC protocol that is used internally for communication between functional units. For transmitting packets the translation is reversed.

*HTAX* [15] is an internal 9x9 on-chip crossbar that connects the three virtual channels of the host interfaces to the functional units inside EXTOLL. It represents a protocol independent switching architecture with low overhead for direct communication between two ports.

### 1.3.2 EXTOLL Core Logic

The *VELO* (Virtualized Engine for Low Overhead) [16] functional unit has been designed to transmit small messages up to a size of 64 bytes (which corresponds to one cache line in today's x86 based processors) efficiently and with minimal overhead. On an FPGA prototype sub  $\mu$ s latency could be achieved with minimum size packets. Nevertheless, VELO can also transmit larger messages, however, its efficiency decreases the larger the packets get.

For larger bulk transfers the *RMA* (Remote Memory Access) [17] unit was introduced. It provides put and get operations that can be started with a single packet from the host system and then work independently of the processor. It can work with both physical and virtual addresses. Both VELO and RMA are fully virtualized and therefore allow concurrent access from different user space processes.

The *ATU* (Address Translation Unit) [18] is a supporting unit for the RMA and provides address translations from virtual addresses that are used in RMA to physical system addresses without involving the operating system and therefore without the kernel overhead

that would be required otherwise. Essentially it can be viewed as an MMU (Memory Management Unit) since it performs a comparable task to the MMU in a CPU.

The *SMFU* (Shared Memory Functional Unit) [19] introduces support for a non-coherent distributed shared memory by forwarding local stores and loads to non-local nodes. This is done by splitting the local CPU's address space in local and remote memory. Loads and stores that are mapped to remote memory will be handled by the SMFU and transferred to the correct node and executed there without any additional software support.

The *RF* (Registerfile) [18] module contains control information for all internal units and is also used to collect debug data that is generated in the various instances. Since it is mapped into the system's address space the values can be easily changed through the system software. The register file is automatically generated from an XML specification which leads not only to a RTL implementation but also code for kernel drivers, documentation and verification is generated automatically. With tens of thousands lines of codes and more than thousand internal registers it is an integral part of EXTOLL. Because the register file is organized in a hierarchical way register file nodes can be distributed over the chip and move into functional units to shorten signal paths between the register file and the unit that is controlled by it.

The *SNQ* (System Notification Queue) [18] serves two purposes. It is used to send interrupts to notify the host system that an important event has occurred in the device that should be handled by the kernel driver. The second purpose is to collect debug information upon triggers from units in the design and dump the information to main memory.

The *Debugport* [20] is actually not a part of the EXTOLL functionality but a safeguard against mistakes in the design phase that might be discovered after the chip is produced and brought up in the lab. It is also, as the name implies a method to access debug information after the host system has crashed and software access to the device is no longer possible. In order to perform this functionality the module contains two external interfaces. The Debugport connects to an SPI flash that can contain replacement values for registers in the register file. Those values will be loaded into the chip before the internal reset is released so that they are available at the startup of the logic. By creating this bypass it is possible to correct wrong initialization values for the host interface, for example, that might keep the chip from being recognized in the system. In order to load debug values from the register file without access to the system the Debugport also contains an I<sup>2</sup>C slave that is able to access the complete register file. The module shares a direct access path to the register file with the SNQ so that it does not have to depend on a clean state in the rest of the system.

## Introduction

---

The *NP* (Network Port) [14] is the interface between the functional units and the EXTOLL NIC. Since the functional units are not aware of protocol details the required framing and deframing is performed in the NP as well as the flow control required for the network.

### 1.3.3 EXTOLL NIC

The key element of the EXTOLL NIC is the internal 11x11 *EXTOLL XBAR* [14]. The routing is performed on a table-based algorithm [21] which makes it easy to implement adaptive routing and reroute packets in error situations. Since switching is done using VCT (Virtual Cut Through) the crossbar has significant memory requirements. With a route-through latency of 17 clock cycles it shows an excellent performance.

The *LP* (Link Port) [22] protects EXTOLL packets by adding a CRC to them. In the case of an erroneous transmission the packet is marked and a retransmission is started. Therefore the LP contains a retransmission buffer that is able to store at least as many packets as are required to perform a retransmission from the point when the defective packet was sent.

The *Link Control* units are the interface between the logical packets in the link port and the physical transmission in the PHY layer. Like earlier FPGA based versions the EXTOLL ASIC uses 8b10b coding to ensure DC balanced transmission. The Link Control module also contains logic for link initialization, rate switching and training.

## 1.4 Outline

This thesis can be divided into two parts. Chapter 2 and 3 focus on the EXTOLL ASIC and its implementation while the subsequent chapters 4 and 5 deal with the ecosystem around EXTOLL. Each chapter focuses on one key aspect in the design that was analyzed and optimized during the design process.

Chapter 2 gives an overview of the technological obstacles during the planning phase of the EXTOLL chip. It also shows the relevant changes from an earlier FPGA based implementation to an ASIC based implementation and focuses on the technological differences. Some modules that were specifically designed for the ASIC implementation are also introduced.

Chapter 3 introduces a modern ASIC flow that is used to implement the EXTOLL chip. It gives a short overview of the complete design flow starting with synthesis up to the final tapeout for chip production. A set of guidelines are presented for collaborative work on such

a large chip. The rest of the chapter focuses on the development and optimization of the floorplan with respect to the package layout developed in the next chapter.

Chapter 4 analyzes the design space for the package and PCB development for EXTOLL and introduces a suitable solution while keeping signal integrity, production cost and other relevant factors in mind.

At last, chapter 5 focuses on the testing of the EXTOLL ASIC for defects. It describes the challenges in wafer testing and analyzes the cost structure of a complete chip test process. In the end a test environment is presented that is able to perform these tasks in a lab environment.

The thesis is concluded by a last chapter that summarizes the results that were worked out in this thesis. It also reviews the design process and its complexity and introduces a set of guidelines that were learned throughout the development. Finally, it also gives an outlook to future developments regarding EXTOLL.



## 2 EXTOLL ASIC

### 2.1 Design Decisions

The EXTOLL research project has been started as a successor of the ATOLL interconnect, the previous interconnection network of the Computer Architecture Group at the University of Mannheim. After the first version of EXTOLL (R1) was able to confirm and supersede [23] the competitive results of ATOLL and more features were introduced to distinguish the network from other vendors it was decided to move EXTOLL from a sole research interest to a real product to be introduced to the global HPC market. Although ATOLL was built as an ASIC, subsequent development switched to FPGAs that could be utilized easily as a rapid prototyping platform. The idea to commercialize EXTOLL had to lead to a review of design alternatives in order to determine the best path to create a product with a profitable price structure and competitive performance.

#### 2.1.1 Performance Comparison

In order to define the minimum performance parameters that EXTOLL had to achieve a look at the competitors at that time was necessary:

Product	Technology	Clock Frequency	Link Bandwidth	Latency	Msg rate
<b>Infiniband QDR</b>	ASIC	~500 MHz	40 Gb/s	1.59 $\mu$ s	6.7m msg/s
<b>10GE</b>	ASIC	~125-312 MHz	12.5 Gb/s	12.5 $\mu$ s	<2.5m msg/s
<b>Cray Gemini</b>	ASIC	650 / 800 MHz	75 Gb/s	1.5 $\mu$ s	~2m msg/s
<b>Tianhe-1a</b>	ASIC	unknown	80 Gb/s	2.5 $\mu$ s	1-3m msg/s
<b>TOFU</b>	ASIC	312.5 MHz	50 Gb/s	1.5 $\mu$ s	>8m msg/s
<b>Ventoux (EXTOLL)</b>	FPGA	200 MHz	16 Gb/s	1.2 $\mu$ s	25m msg/s

Table 2.1 Competitor Overview

Three characteristics can be used to qualify the performance of an interconnection network:

- **Latency**

The latency describes how much time it takes to send the smallest possible packet between two communication partners. Unlike the round-trip latency this measurement excludes the time that is spent processing the packet in the receiver and is therefore more accurate.

- **Link bandwidth**

The link bandwidth defines the raw speed of a link between two endpoints including the protocol overhead. This is an important metric for bulk transfers since best utilization is usually reached by using maximum sized messages to carry the data

- **Message rate**

The last important characteristic is the message rate. It defines how many messages can be exchanged between two endpoints in a second. Obviously, the highest message rate can be achieved by using minimum sized packets.

Naturally, all these parameters are interlinked in some way and an overall outstanding performance can only be reached by tuning each characteristic. Nevertheless, some applications will benefit more from improving one metric over another because of their unique communication patterns. All these measurements can be determined by a set of micro benchmarks from Ohio State University, the OSU Micro-Benchmarks suite [24].

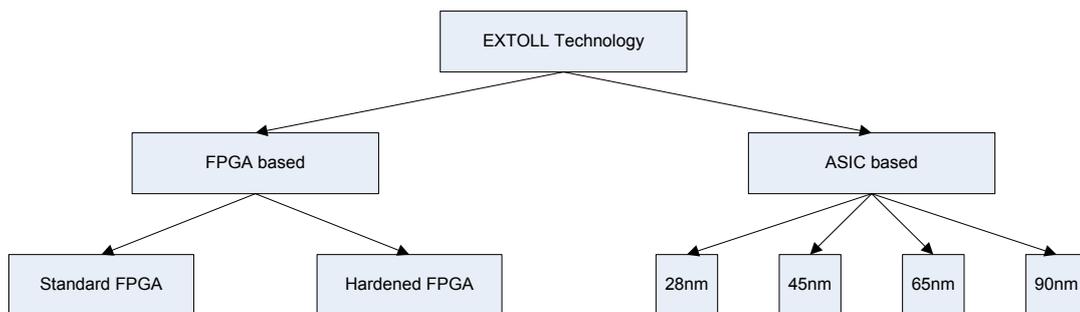
It is clearly visible from the table that the Ventoux prototype is comparable in terms of start-up latency and even surpasses competitors regarding the message rate. The notable exception is the raw link bandwidth which is the second worst of all compared technologies. It is obvious that this is the area in which a commercial product must step up a notch in order to provide unique performance across all metrics. Of course, it is not unfavorable if the other numbers also improve by switching to another technology.

### 2.1.2 Technology Analysis

There are some limitations you have to consider when comparing to an industrial project, especially with regards to development costs. Since EXTOLL stemmed from a research project it was not required to develop the design from scratch. However, the high costs associated with modern semiconductor technology prevent both complete pre-development in a research context and also personal funding. This means that external funding must be acquired from investors or venture capitalists. Unfortunately, this is rather difficult in Germany where investors prefer spending their money on software projects. It is unusual to acquire substantial sums for a hardware project because of the high risk involved. This also

means that the budget for a chip implementation is limited and it is important to earn money as soon as possible in order to keep the business going.

The following design space diagram in figure 2.1 outlines the different paths that can be taken in order to realize a commercial EXTOLL product.



**Figure 2.1 Design Space Diagram for EXTOLL Technology Selection**

FPGA's logic density has steadily increased from generation to generation. Altera's Stratix V family contains devices with up to 950k equivalent logic elements while Xilinx's biggest Virtex-7 device even offers two million logic cells. Although these numbers are not exactly comparable because of different internal structures they both show that FPGAs provide an enormous amount of logic resources. As a comparison, the EXTOLL Ventoux prototype implemented on a Virtex-6, albeit with a slightly lower logic complexity, fills up a device with 240k logic cells. Thus, it can be assumed that an implementation of all the logic inside a single FPGA would be feasible. However, prices from an authorized distributor for the largest Virtex-7 available are in the range of \$30,000 per device<sup>2</sup>. Although Xilinx offers significant discounts for larger volumes [25], the final unit price would probably still be more than \$1,000. Coupled with the fact that it is very hard to exceed internal clock frequencies of 200 MHz (as seen in the prototypes) for complex logic it is easily transparent that it is not possible to build a product that is competitive in both performance and financial aspects.

Both Altera and Xilinx as major FPGA vendors also offer solutions for hardened FPGAs. However, their approaches are fundamentally different. Xilinx takes FPGAs that are not 100% functional and checks if the defective area affects the customer's design. If the design does work correctly this chip is sold as an EasyPath device. Since the chips are basically the same devices as the original FPGAs they offer the same features as before but their performance will not increase. The only advantages of this approach are the quick turn-

<sup>2</sup> Avnet price list, checked in 09/2012

around time and a price reduction of about 30% per device [25]. Altera's HardCopy devices on the other hand are custom produced devices. They contain configurable logic elements as basic building blocks similar to the structure of an FPGA as well as the custom IP that is also available in FPGAs like PCIe macros, however, the chips are fabricated by creating masks for the upper metal layers that contain the wiring between the logic blocks. Since the lower metal layers in a custom chip that contain the structures with the finest pitch and therefore are the most expensive masks in an ASIC design are fixed the customer only has to pay for the metal layers required for wiring. Because these structured ASICs are completely different chips than the FPGAs they must be requalified for correct functionality. In contrary to Xilinx's solution HardCopy chips offer an increased performance of up to 50% and a significant power reduction [26]. Altera also promises a price reduction of about 80% per unit compared to a similar FPGA without taking into account the higher NRE costs for the masks. Though a performance improvement is achievable the results do not justify the higher investment.

The most versatile solution is the production of a custom ASIC. This gives the most flexibility with regards to features that can be incorporated into the chip. A short look at the competitor overview in table 2.1 shows that the certain way to reach acceptable performance is the implementation in an ASIC. However, in comparison to FPGA solutions ASICs suffer from high initial NRE costs:

- **Mask costs**

All masks for the ASIC production must be fully paid by the customer. A full mask set will cost hundreds of thousands of dollars even for relatively mature technologies like 90nm. One can say, that the mask costs for each step to a smaller process node will roughly double which sets a full set for 28nm in the range of several million dollars.

- **EDA tools**

While FPGA vendors sell their tools for a moderate price of less than \$10,000 for dozens of licenses or even give the tools away for free designers that want to implement an ASIC need a whole set of tools for the different steps in the design flow and must pay prices that are an order of magnitude higher for a single license.

- **IP costs**

A large number of IP elements (either Soft IP like PCI Express cores or Hard IP like serializers, PLL and memories) that come for free with FPGAs must be purchased for an ASIC development. Although foundries like TSMC usually offer a basic functionality like standard cell libraries, simple CMOS I/O libraries and sometimes

memory compilers for free it might be important to buy these elements also from 3<sup>rd</sup> party vendors if the performance of the free cells does not match the requirements. Depending on the complexity of the IP and the technology node prices range from tens of thousands of dollars up to more than a million dollars per IP license.

Altogether, the initial investment is rather high and exceeds the volume of one million dollars without problems. Despite these circumstances the price per unit is relatively cheap. Depending on the chip size which determines how many chips can be cut out of a wafer a price of less than \$100 per die is undoubtedly achievable.

Since IP costs and EDA tool prices do not change much throughout process nodes the differentiating cost factor are the mask costs. Therefore the correct technology is a combination of technological feasibility (e.g. required performance, IP availability), existing investment and expected revenue.

With a prospective volume of about 25k units over the lifetime of EXTOLL it is very important to keep the costs as low as possible. An investment in 28nm, for example, would lead to a tremendous unit price (especially regarding the rumored low yield) which will move the point of ROI to the end-of-life of EXTOLL which means that no profit can be reached at all.

Because of the low volume, a MLM (multi-layer mask) mask set [27] is a good way to cut costs. For an MLM four layers are combined into one reticle, thus reducing the largest possible die size to one quarter of a reticle. This allows a reduction of the mask costs because fewer masks are needed in total. However, the price of the wafer goes up because it needs more time to be processed due to the additional time required on the wafer stepper. However, for low volume production the mask cost savings exceed the extra cost of the wafer.

For the design a couple of external IP is needed:

- **SERDES**

An overview of serializers that are able to support PCIe Gen3 [28] shows that no IP is available for 90nm which eliminates this option. Designing Gen3 IP at mature nodes seems to be technologically difficult (or unprofitable) because only few providers offer such IP, most offerings for PCIe Gen3 are available at 28nm. Many vendors skipped the 65nm and 40nm process nodes and designed directly for 28nm.

- **Standard cells**

Trial synthesis runs at 65nm showed that the technology is able to support the target frequency of 750 MHz without changes to the RTL (like adding additional pipeline stages). Because 90nm was already out of the question because of the missing serializers it was no longer considered for evaluation. Since 65nm already meets the performance goals smaller nodes will only increase the available margin for increasing the internal clock frequency.

- **HT3 PHY**

The PHY that provides the physical layer for the HT3 core is only available in 65nm and was originally developed by ATI. After AMD had purchased ATI the PHY was donated for use. However, HyperTransport targets only a niche market since AMD continuously lost market shares in the server market over the last few years. Thus, the availability of an HT PHY is no key factor for the final decision.

- **PLL**

A PLL was already available for 65nm from a former research project. A design for other nodes would be possible, the expected performance can be reached in all nodes.

- **I/O cells**

CMOS I/O cells are provided by the foundry and do not have to be purchased. Special purpose I/O cells that are required for differential signaling must be either purchased or developed. However, there is no special performance criteria for these cells so that they can be obtained for all technology nodes.

- **Memory**

Memories are usually the factor that limits the design speed in a digital design. As before, it was determined by evaluation that the required performance could be reached by using 65nm. Similar to standard cells, smaller process nodes will only improve the speed and increase the margin for running at higher frequencies.

In the end it was decided to design EXTOLL as a 65nm chip because the combination of IP availability, mask costs and expected revenue was the most compelling of all alternatives while also maintaining the performance goals that are needed to outperform the competition.

Consequently, the following subsections will present the challenges and design guidelines for a successful port from an FPGA based prototype to an ASIC solution.

## 2.2 Transition from FPGA to ASIC

EXTOLL is developed by using the Verilog Hardware Description language [29]. The source files consist of more than 280,000 lines of code that utilize roughly 13MB of disk space<sup>3</sup>. In general Verilog is target platform agnostic and will only be mapped during synthesis to a target technology. However, there are certain aspects that need to be handled in a different way depending on the final implementation.

Due to the size of the code base it is not feasible to maintain a completely separate development environment. Instead it is important to keep as much code as possible unmodified and shareable across all platforms.

Nevertheless, there are some constructs that will have to be developed separately while keeping in mind that the overall structure of the design should not be modified.

The following aspects have been identified for special consideration:

- Internal memory, i.e. how the embedded memory is generated, accessed in the chip and how it is protected
- Clocking / Timing, i.e. how are the clocks generated
- Reset, i.e. how is reset generated and distributed
- Testing, i.e. is a special testing methodology required
- Toplevel, i.e. how are the modules connected together

These issues will be addressed in the following paragraphs and guidelines for a successful transition will be worked out.

## 2.3 Design Guidelines

The beginning of the design phase is the place where most impact can be made on the performance of the chip. RTL that was badly implemented will hinder the overall performance and can even waste runtime because tools are forced to optimize sections of code that will have a hard time reaching timing closure or might even fail timing. An old saying in computer science, “garbage in, garbage out”, is also valid for hardware design. Therefore it is important to take care already in the beginning of the design process.

---

<sup>3</sup> Values determined in Oct.'12, excluding external PCIe core

### 2.3.1 Timing Basics

Flip-flops have a set of timing parameters that must be observed so that the flip-flop is not in danger of becoming metastable. The following table 2.2 and figure 2.2 give an overview of the different timing parameters for a flip-flop that is triggered on the positive clock edge.

Abbr.	Timing parameter	Description
$t_{su}$	Setup time	The time that the new value on the D input must be stable before the rising clock edge
$t_{hd}$	Hold time	The time that the value on the D input must be kept stable after the rising clock edge
$t_{co}$	Clock-to-output time	The time between the sampling of the new value on the D input and its subsequent appearance on the Q output
$t_{rec}$	Recovery time	Consistent to the setup time, but related to asynchronous inputs (i.e. reset)
$t_{rem}$	Removal time	Consistent to the hold time, but related to asynchronous inputs (i.e. reset)

Table 2.2 Timing Parameters and Their Meaning

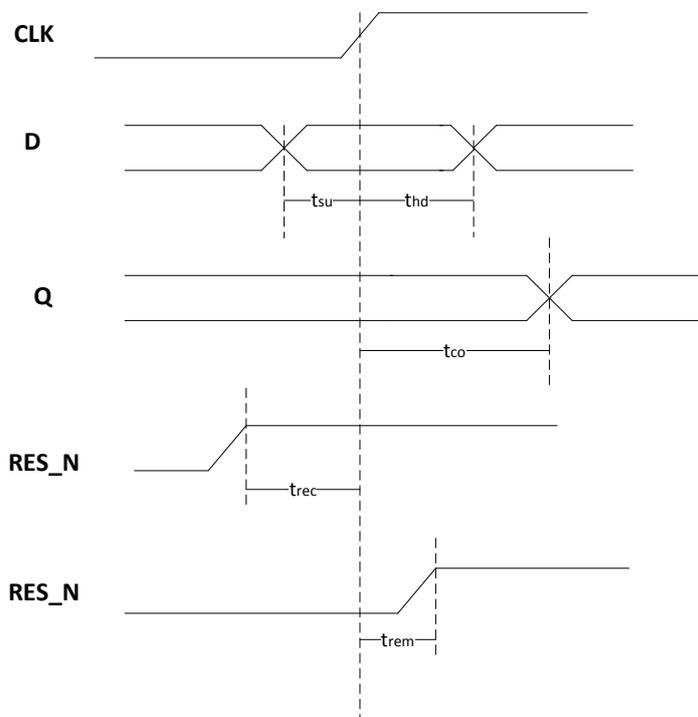


Figure 2.2 Flip-Flop Timing Parameters

These parameters also have an influence on the logic that follows afterwards and the performance that can be reached. Setup time violations can be fixed by reducing the propagation delay of the combinatorial logic before the flip-flop (either by upsizing cells or rewriting RTL) or decreasing the internal clock frequency. Hold time violations are independent from the clock and can only be fixed by buffer insertion or cell downsizing and will lead to a non-functional chip if they are not corrected before tapeout.

For the setup time the following formula is valid where  $t_{clk}$  is the clock cycle time and  $t_{pd}$  is the propagation delay of the logic between two flip-flops.

$$t_{clk} \geq t_{co} + t_{pd} + t_{su}$$

Equally, the hold time is defined by the following formula:

$$t_{co} + t_{pd} \geq t_{hd}$$

### 2.3.2 Reset

A global reset signal is used to initialize a design to a well-known state. However, FPGAs employ a completely different reset strategy than ASICs. FPGAs load their configuration from an external bitstream which means that everything in an FPGA has a predefined initial value so that a reset is actually not required [30]. Nevertheless, the internal registers inside the FPGA support both asynchronous and synchronous reset styles. Analysis show that a completely synchronous reset results in a better timing, an observation that is also indicated by Xilinx [31]. Therefore, the EXTOLL FPGA prototypes use synchronous resets throughout the complete design.

ASICs in return must be reset so that all registers have an initial state. Synchronous resets suffer from two disadvantages. They require a stable clock because the reset signal is only sampled at a clock edge and the reset pulse must be at least one cycle long so that it can be detected. Therefore ASICs are usually reset asynchronously. However, this introduces the problem of reset removal as characterized in the previous section 2.3.1.

A well known solution is the reset synchronizer circuit [32][33] showed in the following figure 2.3.

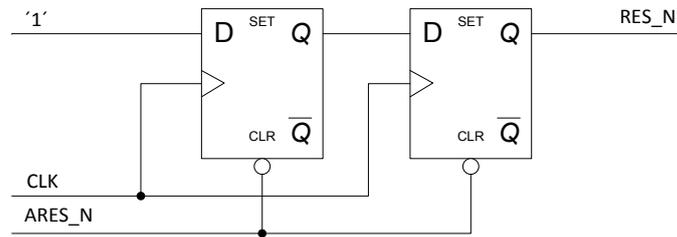


Figure 2.3 Reset Synchronizer Circuit

This simple circuit leads to an asynchronous assertion of the reset signal because the  $Q$  output of the second flip-flop (the final reset signal) will go to  $0$  immediately as soon as the  $ARES\_N$  signal switches to  $0$ . In the case of reset deassertion the value  $1$  of the first flip-flop's  $D$  input will be propagated through the two stages. Only the first flip-flop in the synchronizer is in danger of becoming meta-stable because the removal time might be violated. However, the second flip-flop will sample a valid input in the first cycle. In the second cycle the input will be stable again and once again a valid input will be sampled. Thus, the second flip-flop cannot become meta-stable and an invalid state will not be propagated along the reset tree.

Because of the arbitrary reset styles all *always* blocks have two sets of sensitivity lists that are distinguished by a global *ifdef* macro. One set will lead to a synchronous reset style for an FPGA implementation the other will infer an asynchronous reset for an ASIC.

In general it is important not to confuse a reset signal with an init signal to set the logic back to a known state sometime during operation. Reset should only be used directly after power-up to avoid unknown states. If it is necessary to “reset” the logic later on this functionality should be introduced into the synchronous logic path. Adding combinatorial logic to the reset path will only lead to problems later on during construction of the reset tree in the backend design process. It might also introduce the problem of a reset glitch because the output of a combinatorial cell might change its state for a short time (i.e. it glitches) whenever an input changes even though the change on the input will not lead to another output. In this case the following logic might be reset unintentionally.

There is also a pitfall with the Verilog implementation if registers are used inside the logic although they are not resetted explicitly. This will cause the synthesis tool not to treat reset as an ideal net but like any other net in a synchronous design. The condition for the non-reset case (that is present in the code) will be combined with the logic that feeds the register. This leads to a highly buffered reset net and completely ruins the timing for that path.

## 2.4 Embedded Memory

In FPGAs using memory is very easy. All proprietary synthesis tools from either one of the major FPGA vendors or 3<sup>rd</sup> parties like Synopsys recognize simple Verilog language constructs and are able to map them to the integrated memory blocks [34] available in the FPGA. This flow is depicted in the following figure 2.4.

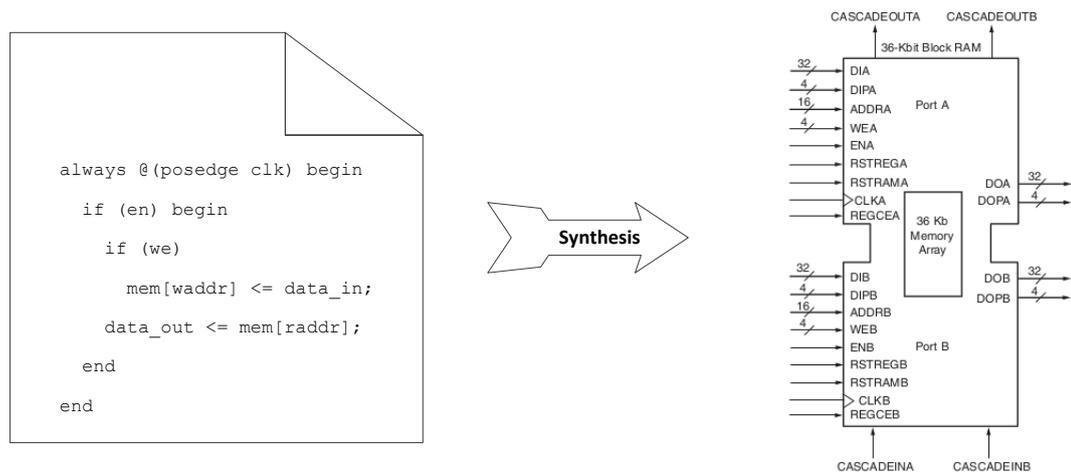


Figure 2.4 FPGA Memory Flow

In ASICs, however, memories are usually generated with a memory generator, a special purpose tool that builds the memory already as an optimized structure for the target technology. The designer must instantiate this customized block in the design to use it. This gives less flexibility in terms of configuration.

EXTOLL consists of about eighty different RAM configurations, ranging from small sizes like 16x61 up to really large memory sizes like 4096x64. Although it is possible to generate RAM macro blocks for each of these configurations this would result in a large amount of libraries that only differ marginally with respect to area and speed and increase the load time and the memory consumption of the EDA tools.

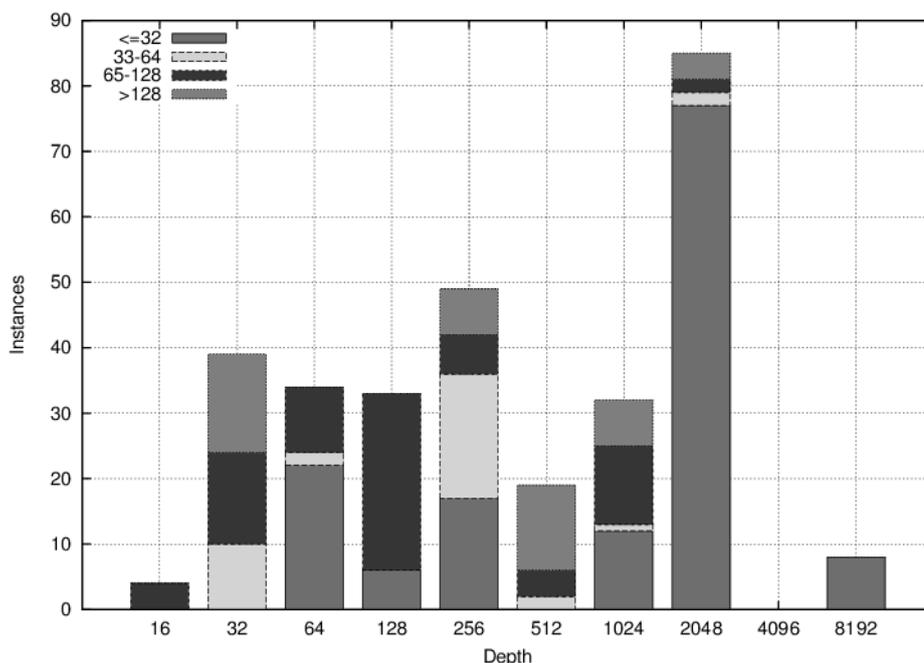
Therefore the best approach is to generate a small number of building blocks in typical configurations and build up all other configurations out of these building blocks. Although this will waste area for RAMs that cannot be matched exactly to these blocks the easier handling outweighs that disadvantage by far. For area critical designs this conclusion is probably not valid, the EXTOLL die, however, will have enough space so that resource saving

is not necessary. Constraining the designer to use only a small set of memory sizes as an alternative only limits the flexibility in handling RAMs. It makes portability more difficult since the basic RAM block size of an FPGA is not very efficient for an ASIC due to its small capacity.

The following figure 2.5 shows the distribution of memory sizes that are required. For simplicity the widths are divided in bins of powers of two. Several observations can be made from the chart:

- There are a large number of RAMs with widths of less than or equal 32 bits that have many entries though. Altogether almost 100 instances can be found.
- Combining the first four columns will add up to another 100 instances. Most of these instances are wider than 64 bits.
- Most of the RAMs with a depth of 256 entries are less than 64 bits wide.
- Instances with widths of more than 128 bits are relatively uncommon.

Since the internal datapath has a width of 128 bits it is evident that there will also be many instances required that have the same data width. Many modules employ at least one FIFO to store internal data before it is forwarded to the next unit.



**Figure 2.5 Memory Size Distribution**

An analysis of the speed of the memories which is shown in the following figure 2.6 (y-axis is unlabeled to hide proprietary data) shows that the speed decreases in a linear way if the RAM becomes wider. It also shows that the starting speed for the least wide configuration decreases with an increasing number of entries. While the distance between the three smallest configurations is almost the same, the offset becomes higher as soon as 128 entries are exceeded. Although all configurations are nominally faster than the required internal clock speed of EXTOLL the margin between that frequency and a RAM with 256 or more entries might become critical so that there might be problems getting timing closure.

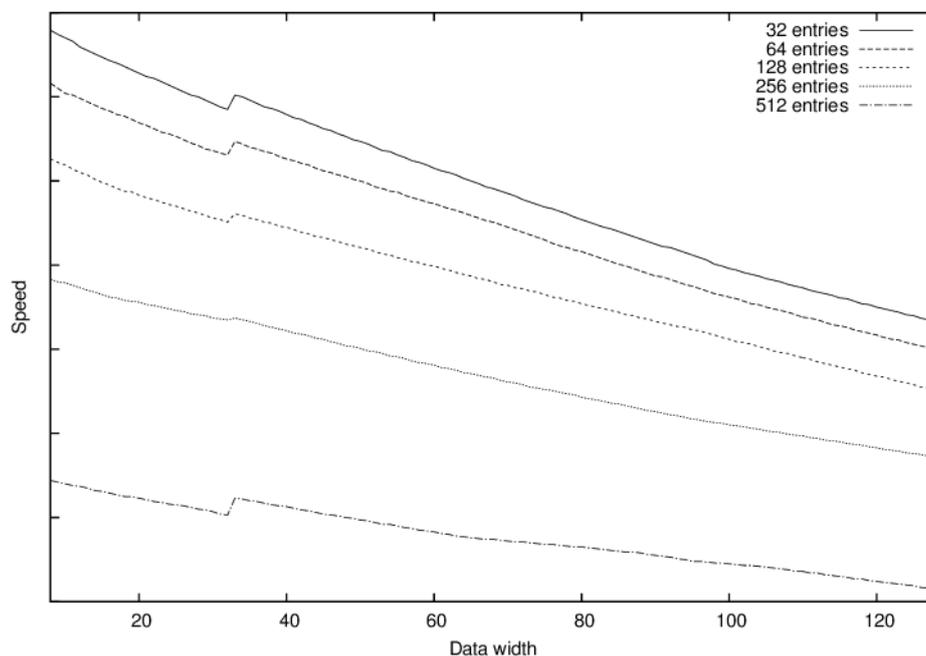


Figure 2.6 Memory Speed Analysis

### 2.4.1 Memory Faults

Memories are more susceptible to defects than standard logic cells because of their optimized, highly packed structure. Memory cells are designed with minimum geometry devices in order to achieve a high memory size / area ratio. Of course, this also means that the probability of a memory fault rises as more and more memories get added to a chip design.

As seen in the following figure 2.7 memories take up a large amount of the occupied chip area in EXTOLL. Therefore it is essential that the possibility of a memory fault must be taken into account and handled in a transparent way.

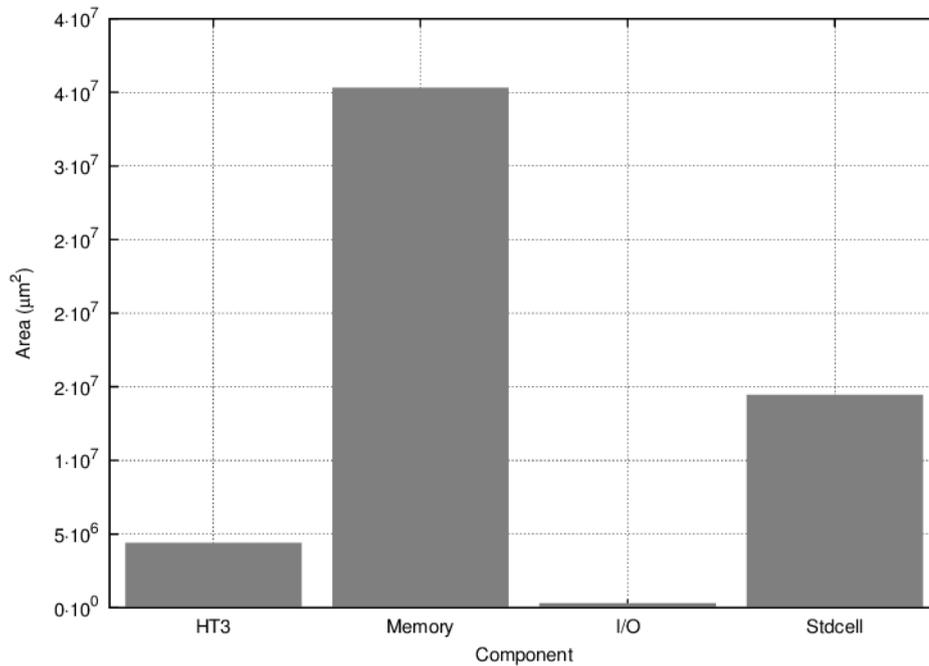


Figure 2.7 Occupied Chip Area

Memory faults can be classified in dynamic and static faults. Permanent faults are caused by manufacturing errors and can appear with many different fault models. While faults in the address decoder might make some rows or columns in the memory array inaccessible, faults in the array itself might exhibit diverse effects. For example, the cell could be stuck at either 0 or 1 (so called stuck-at faults), it could lose its value (data retention fault) or it might be influenced by the activity of surrounding cells (coupling fault). These faults can be discovered by running BIST (built-in self test) algorithms on the memory [35] which then leads to a removal of the device from the production lot.

Transient or intermittent faults, so called soft errors, are usually caused by external influences like radiation particles and can cause one or more cells to flip their value.

Both types of faults are well known and can be handled by introducing redundancy. Introducing spare rows or columns [36] is a favored way for handling manufacturing defects. If a row or column is identified as defective during BIST all accesses are remapped to the redundant cells inside the memory. The mapping itself is not done dynamically at each startup, instead the reconfiguration is permanently stored in the memory by using electrical fuses. If a process is known to have a bad yield this is a great way to get a larger number of fully functional chips from a wafer, however, the mapping and the additional logic inside the

RAM block leads to a significant degradation of the RAM's performance that can be in the range of several hundred MHz.

Transient faults can be handled by introducing logical redundancy for on-the-fly error checking and correction (ECC). The complexity of this operation depends on the failure rate that must be covered by it. Regardless of the selected algorithm there will be a penalty on the timing of the RAM and additional area consumption. A small side effect is that the probability of a fault is slightly higher because the size of the RAM will also have to increase in order to store the redundancy information. A study [37] by Intel suggests that 75% of all defects can be corrected by using a SECDED (Single Error Correction, Double Error Detection) algorithm while upgrading the error correction to DECTED (Double Error Correction, Triple Error Detection) will improve this rate only marginally by 2% with the drawback of a more complex implementation. A similar conclusion can be found in an analysis in which RAMs are exposed to heavy ion radiation [38]. With a low dosage (which is the most applicable scenario for normal operation) single bit upsets dominate the evaluation of encountered bit errors. The same paper also compares the complexity of SECDED and DECTED algorithms with the conclusion that the encoder complexity is almost the same for a SECDED and DEC algorithm, the decoder, however, suffers from a performance decrease of about 50% for a DEC algorithm in comparison to a SECDED implementation. A complete DECTED implementation even takes up to three times longer. With these results in mind a SECDED algorithm is the best solution for EXTOLL.

Because multi-bit error correction is not required, suitable codes like BCH (Bose-Chaudhuri-Hocquenghem) [39][40] or Reed-Solomon [41] are not needed. Instead a simple Hamming code [42] or an improved version SECDED algorithm, the Hsiao code [43] is used to handle single bit upsets.

Depending on the number of available parity bits  $m$  a message with a length of  $2^m - m - 1$  bits can be protected from single bit errors. In order to distinguish between a single bit error that is correctable and an uncorrectable double bit error an additional parity bit is required. Thus, a SECDED implementation requires a Hamming distance of 4.

The following figure 2.8 shows the number of gates required in relation to the usable data in a memory as well as the propagation delay through this ECC circuit. As expected, timing becomes worse for larger data widths. Both encoder and decoder roughly take the same time to perform their task so that the penalty is the same for reading and writing to a memory. In contrast the required area for the encoder does not increase as much as the area of the

decoder for higher data widths. For the highest data width that was analyzed the decoder consumes about three times the area of the encoder.

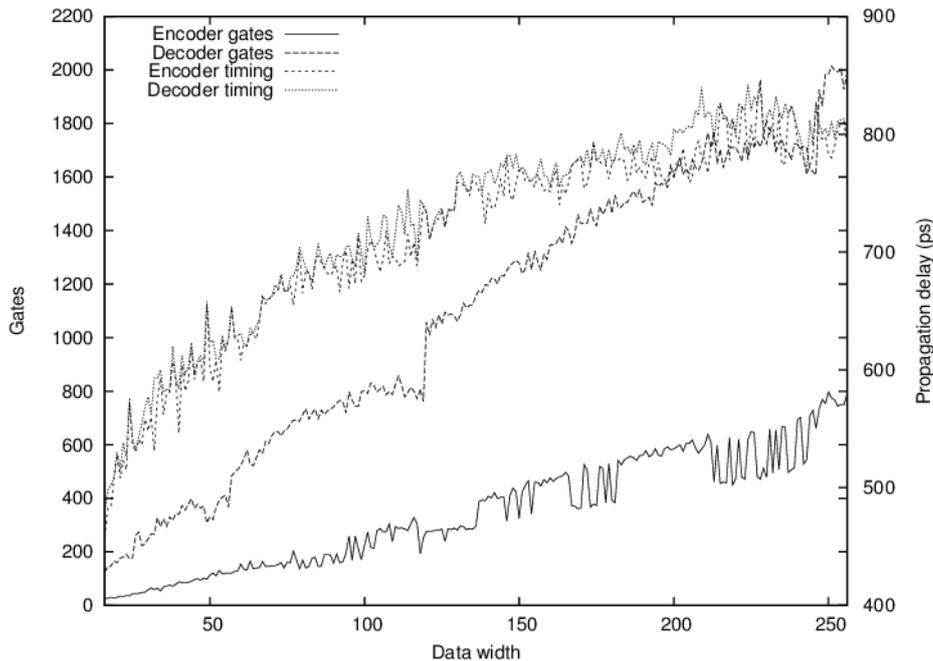


Figure 2.8 ECC Complexity Analysis

As a conclusion it can be said that adding an error correction algorithm that can handle single bit upsets is essential for reliable operation if RAMs are required in a design. Depending on the manufacturing error a SECCED implementation might also give the possibility to reuse chips that would have been marked as defective otherwise, albeit with losing the ability to correct additional errors that might occur. Nevertheless, these chips could be used for internal testing while chips that are fully functional will be shipped to customers.

#### 2.4.2 Automatic Memory Generator

Since it is not feasible to instantiate the correct RAM instance in every module it was necessary to support a parametrizable flow for flexibility reasons. The final mapping to the RAM building blocks should be non-transparent to the designers. This is especially the case for FIFOs that are usually used with parameters for both width and depth, but also many other modules in EXTOLL can be changed with parameters which would result in different RAM configurations. In order to keep the underlying changes invisible several steps are needed.

At first, a RAM building block for every single RAM configuration must be built. As mentioned before only a small number of core blocks is available, all other configurations must be built from these blocks. Because manual generation is both error-prone and tedious an automatic approach was selected. A Perl [44] script takes the desired width and depth of the RAM, analyzes which core block is most suitable and generates the Verilog code for the resulting RAM. The following figure 2.9 shows the internal structure of a selected RAM configuration.

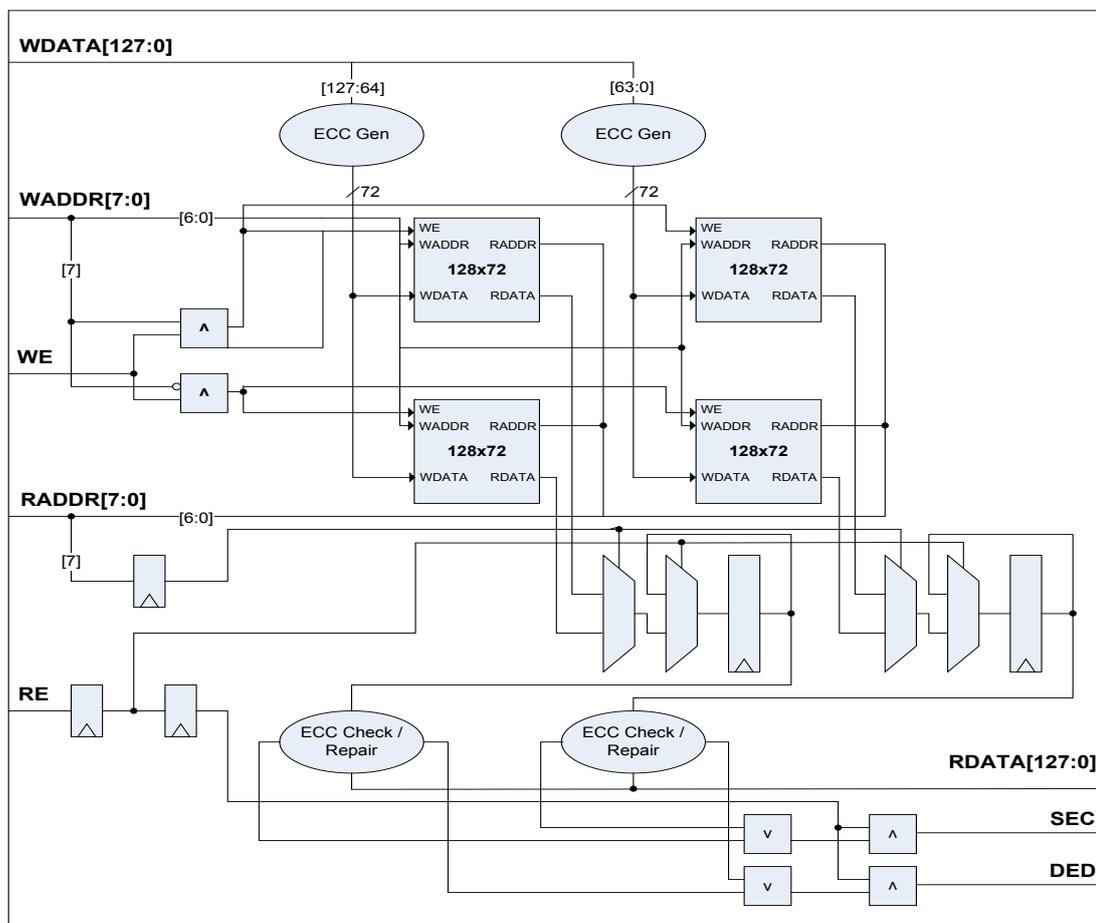


Figure 2.9 256x128 RAM Instance

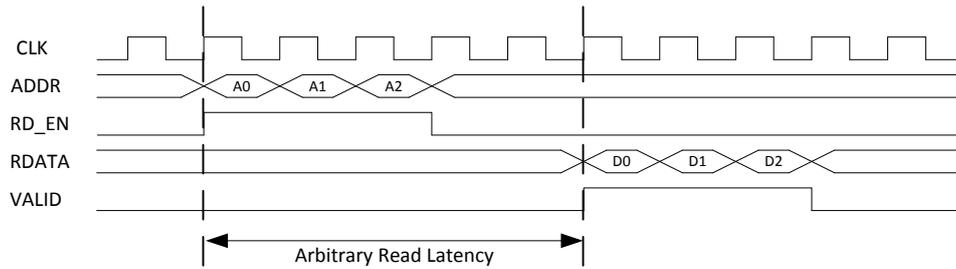
As stated in the previous paragraph ECC functionality is essential. Thus, the best approach is the inclusion of ECC generation as well as ECC checking and error correction in the RAM instance itself. This means that on the interface level the RAM looks standardized with a read port and write port while all the ECC handling is hidden inside. In figure 2.9 a 256x128 configuration is built from 4 single 128x72 core blocks. 128 bits of data that will be written to

the RAM are split in two 64 bit wide streams that are separately protected through ECC. This adds additional 8 bits per data stream so that in the end 144 bits will be written. Because the core blocks only have a depth of 128 entries instead of 256 a second row of RAMs had to be added. The lower bits of the write address select the correct entry in the core block while the highest bit of the address together with the write enable signal select the row to which a new entry will be written.

A read operation is performed in a similar way, but it requires a little bit more logic. The lower bits of the read address are applied to all core blocks simultaneously. The highest bit of the read address which is delayed by a register stage to compensate for the read latency of the RAM controls a multiplexer for each column that selects the correct row that must be read. This multiplexer is followed by a register stage which is only updated if the read enable signal is asserted. This allows the RAM to retain the value of the read port even if the applied read address changes although no new read cycle was indicated through the read enable signal. Each 72 bits wide bus is then fed through the ECC check and repair logic. The resulting 64 bit data streams are then concatenated to the 128 bit read data port. The ECC logic also gives two indicators called *sec* (single error correct) or *ded* (double error detect) that can be connected to the register file, for example, to show that an error condition occurred during operation.

From a timing perspective some special characteristics have to be considered. Both ECC generation in the write path and ECC checking in the read path take some time as seen in figure 2.8. For writing this means that the propagation delay of the ECC generator must be added to the setup time of the internal RAM blocks. In a well designed logic block this should be no problem because the write data port will be driven by a register. Despite being a bad design style the RAM instance is not output registered. This is caused by the fact that the logic in EXTOLL was designed with FPGAs in mind and therefore optimized for the read latency of a block RAM in an FPGA. An additional cycle of read latency would break the following logic. By not registering the RAM output both ASIC and FPGA RAM blocks have the same read latency. However, the time needed to check and probably correct the read data is time that is missing in the next pipeline stage. Since a large part of a clock cycle is taken up by the ECC logic the next stage can only perform small logic functions without breaking timing. Nevertheless, in most cases this is not a problem because many RAMs are actually used in FIFOs which include some additional register stages behind the RAM to support fall-through functionality. Regardless, for future projects it would be helpful to introduce a handshake mechanism with a read enable signal and a subsequent valid acknowledgement

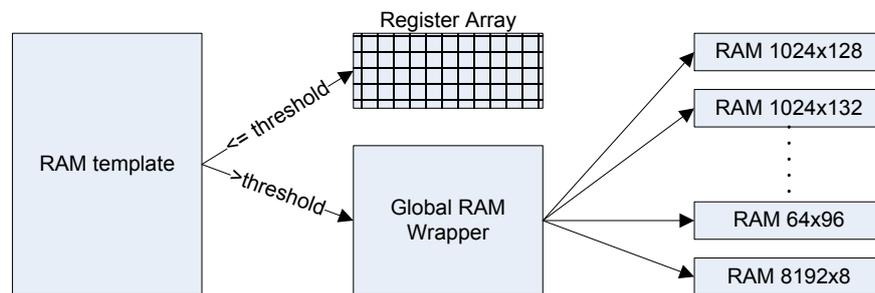
from the RAM as seen in the timing diagram in figure 2.10. This allows the surrounding logic to work without relying on the exact timing parameters of the underlying technology since it now does not have to adhere to a specific latency that is defined beforehand. An additional pipeline stage for ECC checking could have been easily hidden in that case with the result of a much more relaxed timing budget.



**Figure 2.10 RAM Timing with Handshaking**

In a second step another Perl script takes all the generated RAMs and builds a global RAM wrapper for them. This is a parametrizable Verilog module that will instantiate the correct RAM instance depending on the parameter set that is passed. If no matching RAM is found it will throw an error in simulation so that the missing module can be identified easily.

Finally, a set of RAM templates exist for both ASIC and FPGA. They provide an identical interface for both technologies and are used whenever a RAM is instantiated in the design. For small RAM sizes the RAM template does not map the configuration to a set of hard macros as described before but it builds an array of registers which is more efficient for small storage blocks. The threshold can be changed, but is set to a size 1024 bits by default. Altogether, the memory flow contains several levels of indirection as depicted in figure 2.11.



**Figure 2.11 Levels of Indirection**

Three different templates are currently supported:

- **1R1W1C (1 read, 1 write, 1 clock)**  
This is probably the most common configuration because it is usually needed in FIFOs. It consists of separate read and write ports and works with a common clock. All generated RAMs are dual-port RAMs with both a read and a write clock. In order to map the template to the underlying RAM the common clock is simply connected as the read and write clock.
- **1R1W2C (1 read, 1 write, 2 clock)**  
This configuration is mostly needed for transfers between clock domains, commonly in asynchronous FIFOs. It directly maps to the underlying RAM.
- **2R1W1C (2 read, 1 write, 1 clock)**  
This configuration is only used in the register file when both software and hardware must be able to read data simultaneously. Since the RAMs only support a single read and a single write port they cannot be mapped directly. Instead, two 1R1W1C RAM templates are instantiated that are both written at the same time during a write operation. With two separate RAM instances inside that hold the same data a second read port can be offered, however with the penalty of additional area.

RAMs in FPGAs are always initialized and have a definite content. Unfortunately, this led to design mistakes at a few points in the EXTOLL logic that were coded with the assumption that this is always the case. However, RAMs in an ASIC are not initialized and can deliver an arbitrary value. For these cases, a special instance of the 1R1W1C wrapper was designed that performs an initialization of the RAM after the reset signal is deasserted by overwriting each row with a default value. During the time that is needed for this procedure (as many clock cycles as there are entries in the RAM) the RAM cannot be written to or read from. BIST operation will not be impaired by this functionality since it accesses separate test ports and the normal ports are disabled.

Generally, a RAM does not support reading and writing on the same address in the same clock cycle. If it still happens due to badly designed logic, the resulting read data is corrupted and might not reflect the value from the RAM. Logic should be aware of this issue and avoid these situations. Nevertheless, this problem can occur in EXTOLL because some instances are defined with a constant read enable signal and an address overlap will happen sooner or later. For these RAMs the wrappers support an extra parameter that will add supporting logic to perform a RAW (read-after-write) operation. By setting the parameter the write data will be forwarded to the read port in the case of an address violation. Instead of using the corrupt

value from the RAM the data that will be written in the same cycle is delivered on the read port and the hazard situation is avoided. This is also the behavior of an FPGA RAM and will not break the logic around it.

The automatic generation of the memory wrapper and the separate RAM instances is a valuable tool for a more efficient ASIC design. More than 20,000 lines of Verilog code were generated. If an error is found a small change in the underlying Perl code is sufficient and all RAMs are rebuilt with the fix in a matter of seconds. If a new core RAM block was introduced rerunning the tool would automatically check if building a RAM with the new core block is more efficient and rewrite the RAM accordingly.

## 2.5 Design for Test

FPGAs are reconfigurable by design. ASICs on the other side are hardwired and cannot be changed later on. Therefore it is important to make important parameters in the design reconfigurable in the ASIC by moving them to the internal register file so that they are changeable. The stored data can be divided into two categories:

- **Initialization values**  
These are needed to bring up the design to a functional state and cover parameters for the host interface or the internal PLL. A misconfiguration at this stage might cause the chip to be unusable.
- **Debug values**  
Several FSM states, counters and other valuable debug information from each unit are also stored in the register file.

Usually, register file access is done by system software. If the design is not able to bring up the host interface, for example, there is no way to get the system up and running. Therefore it was important to introduce a mechanism to change these values early on. Thus, the Debug Port contains an SPI [45] master that connects to an external flash memory. This flash can be filled with addresses of registers and their updated values. The internal protocol engine makes sure that all relevant registers can be updated before the chip continues its internal operation. By doing this, critical initial values of the design can be fixed.

The Debug Port also includes a second mechanism to access the register file. An I<sup>2</sup>C [46] slave allows external hardware to connect to the register file and read and write its values. There are twofold applications for this. For one, it allows reading debug values even after the

host system crashes which removes the normal operation mode of simply reading the register file from the CPU. This might give invaluable insight to the current state of the design, something which is a lot easier in an FPGA with tools like ChipScope [47] or SignalTap [48]. On the other hand it also allows an external management controller to read status values continuously from the device to monitor correct functionality or collect overall system statistics.

Both mechanisms access the register file via the SNQ interface. In order to offer fair access to both the SNQ and the Debug Port an arbiter was built in front of the register file's SNQ interface to manage concurrent access requests from both units. Since the loading mechanism from the flash is only executed shortly after power-up it will not impair the further operation. The following figure 2.12 [20] summarizes the datapaths inside the module.

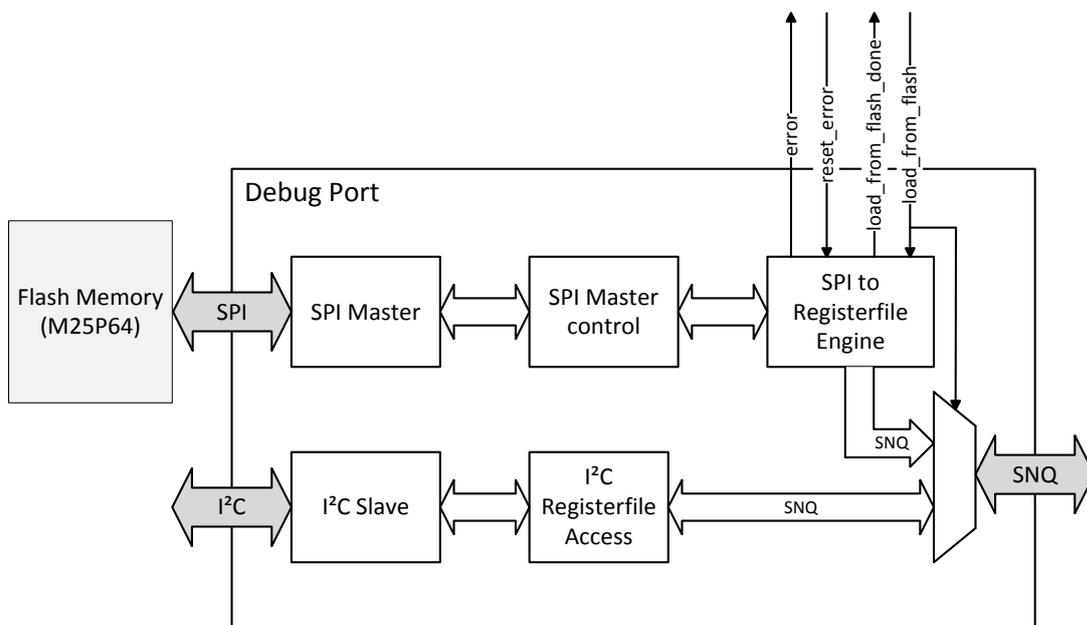


Figure 2.12 Debug Port Overview

## 2.6 Clocking

Clocking is besides reset the most important functionality in an ASIC. For maximum redundancy EXTOLL contains many clocking options so that each stage in generating the system clock can be bypassed and, if required, supplied externally from an on-board clock source like a high-frequency oscillator.

## 2.6.1 Supporting Circuits

There are many cases where a system clock has to be divided down to a slower clock. If the relationship between the two clocks is a power of two this can be easily accomplished with a simple binary counter. The easiest case of a by-2 clock divider is a single flip-flop that inverts its output every clock cycle. Simple odd integer clock dividers [49] have the problem that they do not generate a 50% duty cycle. The following figures 2.13 and 2.15 show two clock dividers used in EXTOLL that will generate a 50% duty cycle together with their respective waveforms in figures 2.14 and 2.16. The correct duty cycle is achieved by generating two waveforms that are phase shifted by using a flip-flop that is triggered on the negative edge of the source clock which are then logically combined to form the resulting waveform.

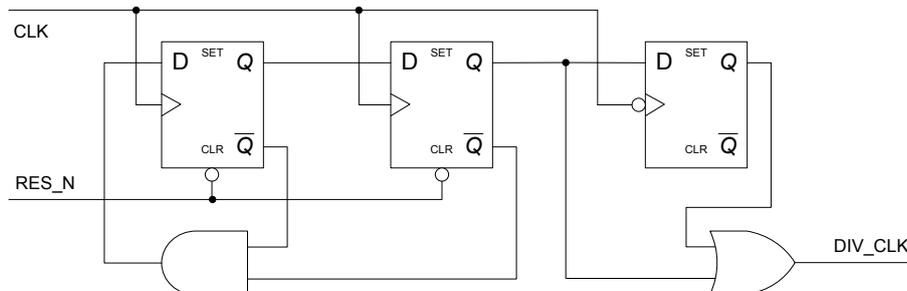


Figure 2.13 Divide-by-3 Clock Divider

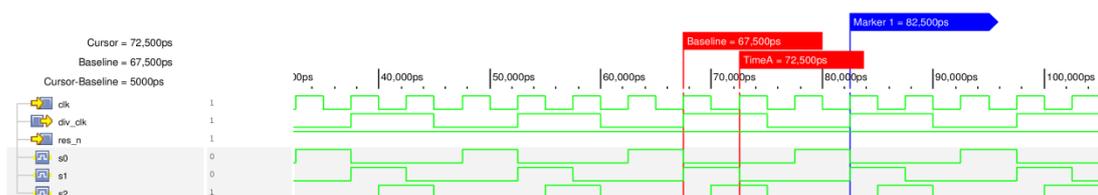


Figure 2.14 Divide-by-3 Clock Divider Waveform

A popular circuit for generating clock odd integer clock dividers is the Johnson counter [50]. It consists of a ring buffer built with a shift register. The inverted output of the last register is fed back to the input of the first stage.



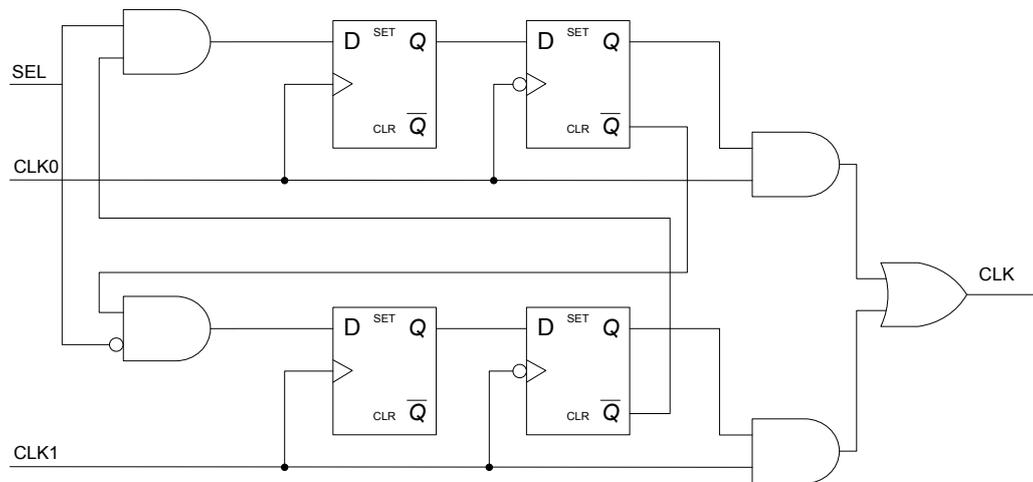


Figure 2.18 Glitchless Clock Multiplexer

## 2.6.2 Clocking Structure

Figure 2.19 shows the internal clocking structure of EXTOLL that is used to generate the main clock. The PLL that is presented in the following section 2.6.3 generates the clock for most of the logic inside EXTOLL. The PLL is fed by a 25 MHz reference clock which is initially generated by an oscillator cell that is connected to an external crystal. In the beginning the PLL is bypassed so that the debug logic can run and overwrite PLL parameters if necessary. Because the PLL does not have a signal to indicate if it is locked a simple counter is used to emulate this functionality. As soon as the PLL is locked the clock is switched without glitches to the higher frequency with the circuit shown in figure 2.18. Afterwards there are two more multiplexers in the clock path before the actual clock root of the design that can be used to supply alternative clock externally. One of these inputs consists of a CMOS input cell to provide the test clock for DFT and is only enabled if the design is run in test mode, e.g. for scan tests. The other input is a backup in the case of a non-functional PLL. Because of the high-speed nature of the clock that must be provided it is designed as a differential CML input.

As soon as the system is up and running there is also the possibility to switch to a synchronous network mode. Instead of the 25 MHz external reference clock one of the seven link clocks that is recovered by the CDR circuit in the serializers and divided down to 25 MHz can be selected as reference clock for the PLL. This switch can only happen if the link is running. Thus, a clock detector circuit is needed to determine if the link SERDES supplies a stable clock. A phase comparator determines if the current and the new reference clock are in phase and switches them indiscernible for the PLL so that the PLL output will continue to

run stable. Of course, as soon as the link clock vanishes (e.g. by removing the corresponding cable) the system must switch back to the on-board oscillator.

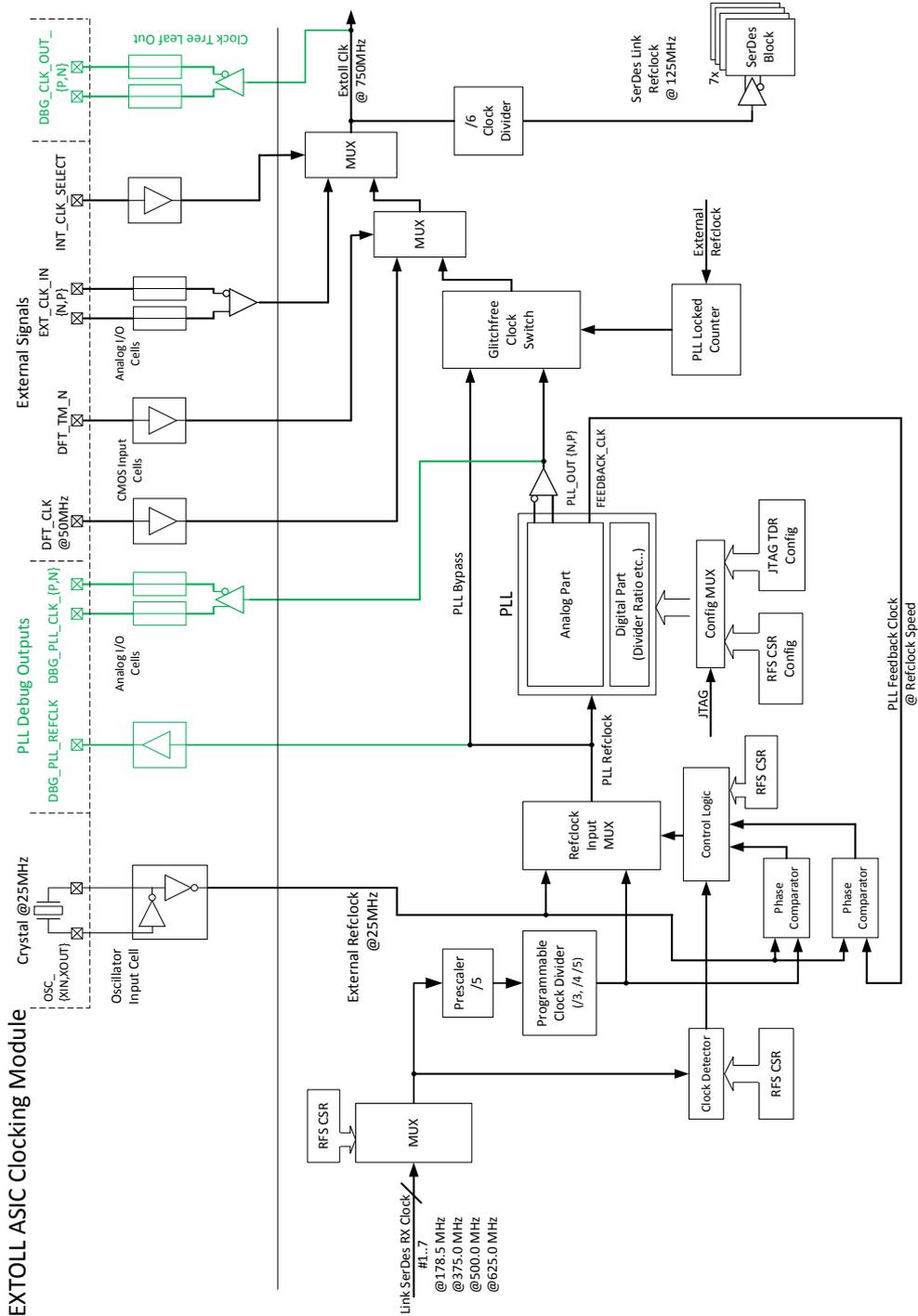


Figure 2.19 EXTOLL Clocking Scheme

### 2.6.3 PLL

The central PLL in the EXTOLL chip was developed by researchers at RWTH Aachen [51] during a collaboration project. The first tapeout was designed for TSMC 65nm LP, however, the PLL was later ported to the GP process at the same node. Its layout is depicted in the following figure 2.20.

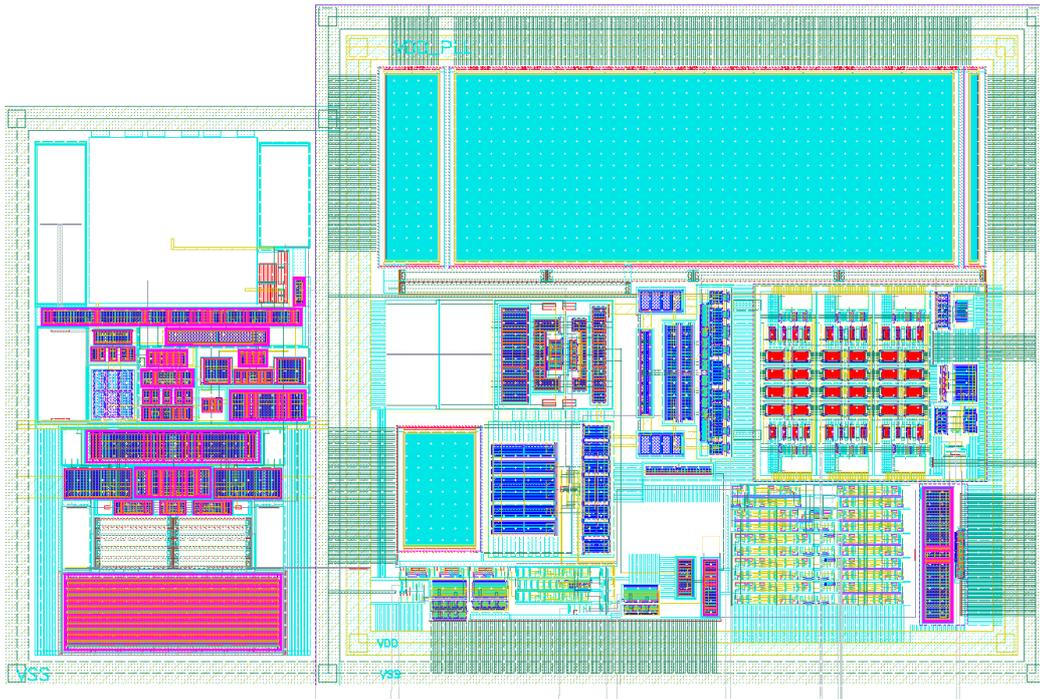


Figure 2.20 EXTOLL PLL

The PLL is fed by a 25 MHz reference clock and can generate output frequencies in the range of 600 MHz to 1.2 GHz. With a simulated output jitter of 400fs at 1 GHz the resulting clock signal is excellent and thereby well suited to be used as reference clock for the serializer circuits. In order to achieve a small footprint it was necessary to avoid using an LC oscillator circuit. Instead a ring oscillator with as little noise as possible was used. The PLL can work both in integer and fractional mode. In fractional mode the divider is split in an integer value of 5 bits and a fractional part with an accuracy of 10 bits. This allows the output frequency to be tuned in very small steps which enables easy overclocking of the final chip, e.g. to rate some dies at higher frequencies.

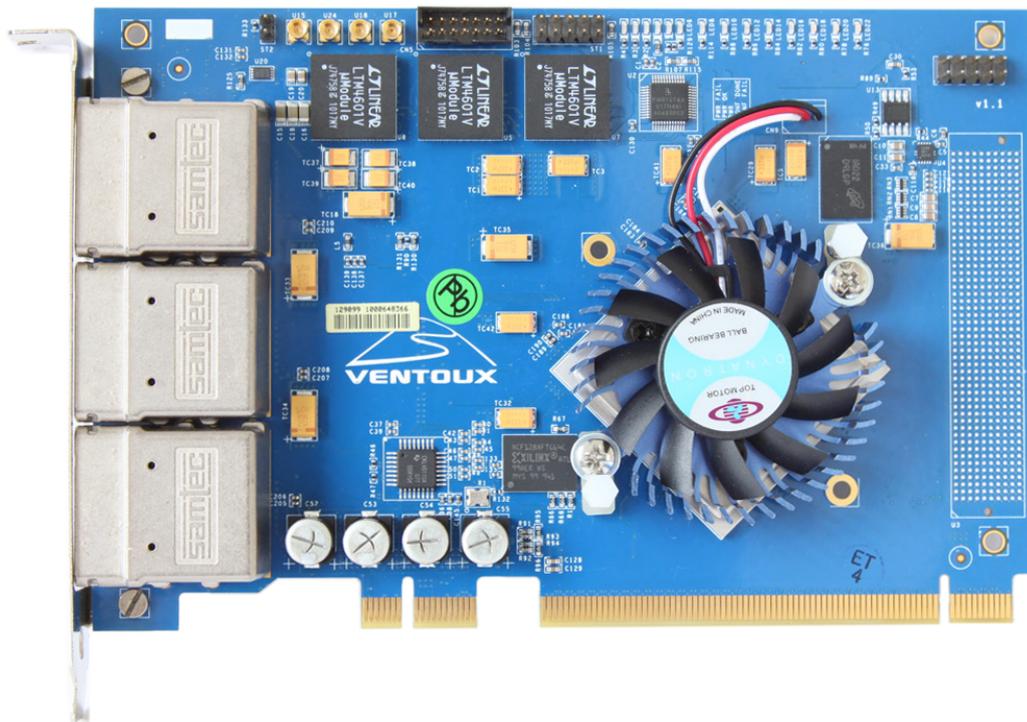
### 2.7 Prototyping

With the increasing complexity of designs verification has become more and more important over the years. Large investments for a production of a chip in cutting edge technologies require confidence in a completely functional design. This means that nowadays a large part of the design process is spent in verification. In order to improve the quality and ease the introduction of verification the UVM methodology [52] was introduced as a standard that is supported by all major EDA vendors and superseded several proprietary implementations.

For EXTOLL a whole set of verification IP was developed in System Verilog [53] to debug single functional units. A complete system-level environment [54] combines these VIPs for debugging the whole EXTOLL design.

Another aspect of verification is the in-system testing. No matter how thorough verification environments will be they will always suffer from performance problems because the simulation speed is several orders of magnitude slower than reality. Also, often in-system tests show peculiar communication patterns (sometimes depending on the host CPU) that were not anticipated when the verification test stimuli were defined. Additionally, the ability to run applications like the Intel MPI Benchmark [55] on a prototype gives confidence that the final hardware will work as expected on typical workloads. Driver development can start early in the design cycle, too.

Since the previous Virtex4 based board [56] that was used to prototype the 1<sup>st</sup> generation of EXTOLL was limited in bandwidth and FPGA capacity it was decided to develop a new board which is shown in figure 2.21. The increased performance of the used Virtex6 device allows an increase of the host interface's speed to HT600. The internal GTX transceivers of the FPGA are routed to the front panel connectors, providing six 4x network links that are capable of running at 6.6 Gbit per lane. Although the device sizes cannot be directly compared because of changes in the internal structure of the FPGA the Virtex6 LX240T device used offers about 3-4 times the space as the previous Virtex4 FX100. Still, the current design is able to fill up the FPGA completely.



**Figure 2.21 Ventoux Prototyping Platform**

Based on the HyperTransport Ventoux board a second prototyping board (Galibier) was developed for PCI Express. The board layout is directly adapted from the previous board which can be spotted immediately by looking at the board in figure 2.22 so that the development effort could be reduced dramatically in relation to a completely new development.

EXTOLL is supposed to be able to run as either a PCIe endpoint or PCIe rootport. While testing the endpoint capability is no problem the second use case is rather difficult since common COTS systems expect devices plugged into their PCIe slots to run as endpoints while the PCIe bridge chipset as opposite party runs as rootport. Thus, a small PCIe backplane which is shown in the following figure 2.23 was developed. This backplane features two PCIe x16 connectors that are directly connected to each other. Required sideband signaling and clocking are provided by the backplane. With this board the Galibier prototype, configured as rootport, can communicate directly with an arbitrary PCIe endpoint device.

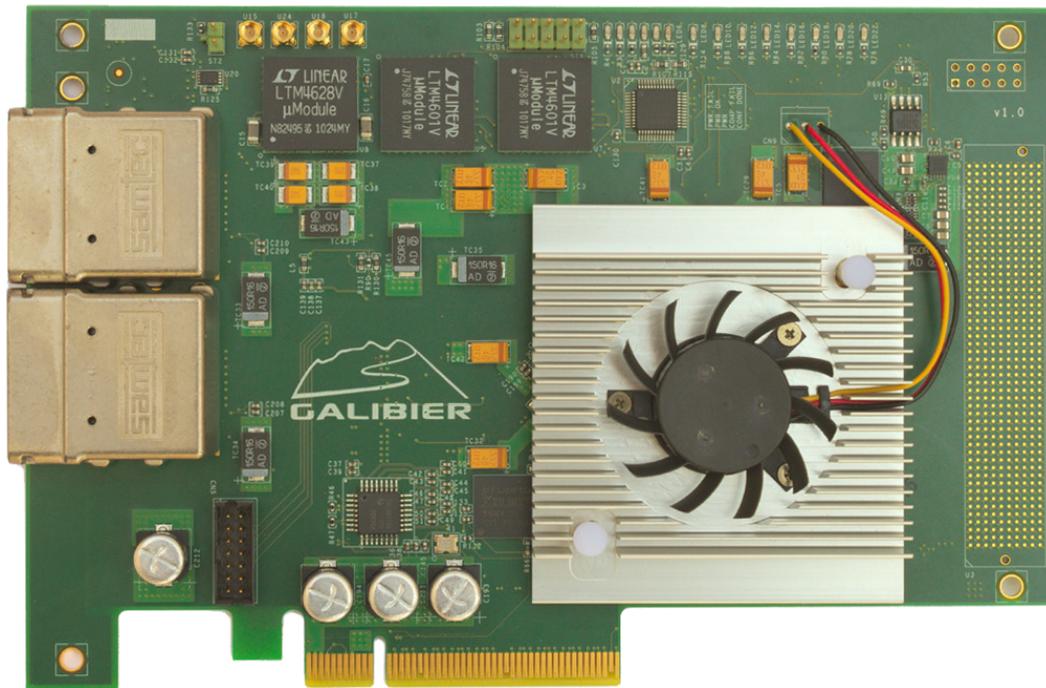


Figure 2.22 Galibier Prototyping Platform

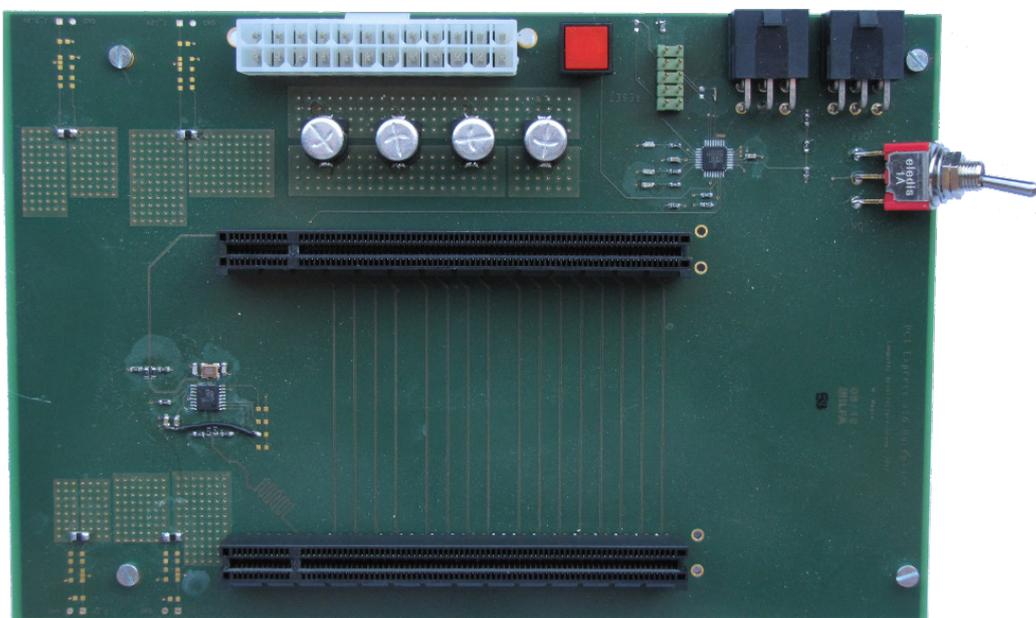


Figure 2.23 PCIe Backplane

At last, in a collaboration project with AMD a HT3 verification platform [55] was developed to explore the option of running HT3 on FPGAs. Since the project never left the proof-of-concept phase and due to severe limitations regarding performance, reusability and stability all development efforts on this platform were discontinued.

## 2.8 Toplevel Organization

For the transition from FPGA to an ASIC development the structure of the toplevel design could not be reused and had to be changed, too.

The FPGA's design hierarchy is depicted in the following figure 2.24.

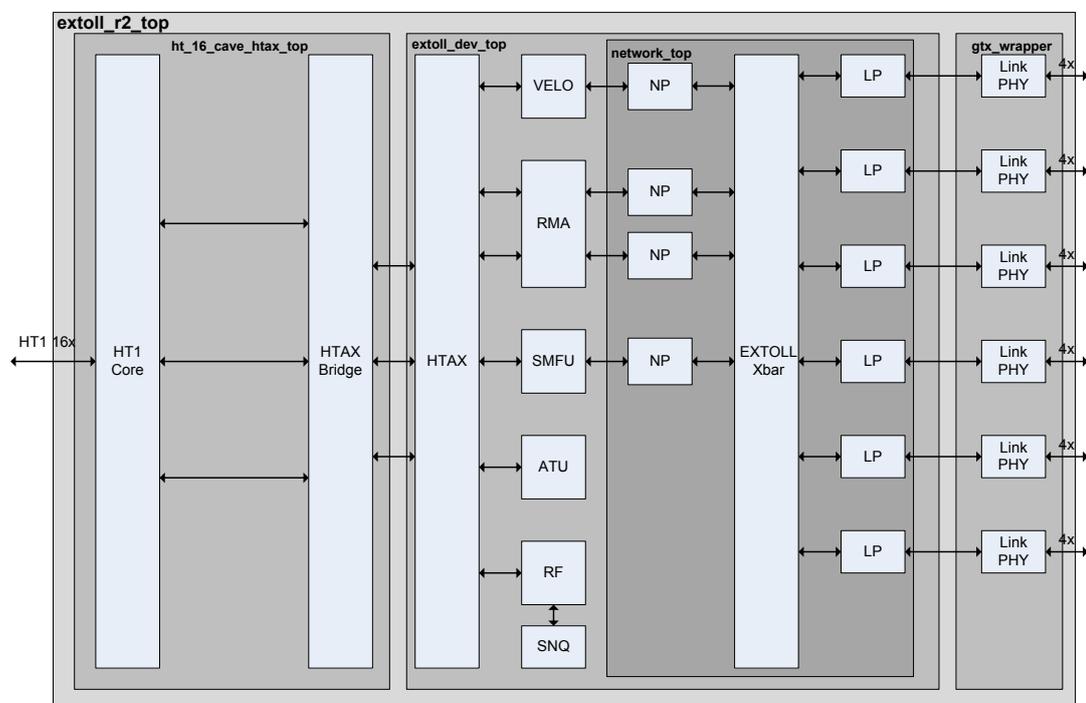


Figure 2.24 FPGA Toplevel Organization

The outer layer *extoll\_r2\_top* consists of three major modules:

- **ht\_16\_cave\_htax\_top**

This module contains the HyperTransport core that is used to communicate with the host system as well as the HTAX bridge. In the case of a PCIe implementation this module is replaced by a PCIe specific module with the same interface signals to the rest of the logic.

- **gtx\_wrapper**  
This module is a wrapper for the FPGA's serializer instances that take care of the data transmissions over the link. This module is technology dependent and specific to a single FPGA generation.
- **extoll\_dev\_top**  
This module encapsulates the EXTOLL logic. It contains another level of redirection. The modules belonging to the network logic are embedded into another submodule, *network\_top*.

Besides these three major blocks the toplevel only contains some glue logic and few small clocking related modules.

For an FPGA implementation the hierarchy in the design is only of little importance besides the logical structuring. FPGA tools do not easily support separate implementations of submodules and rather run the whole design flat through the complete implementation flow. This is partly owed to the fact that the devices only have limited capacity and the design flow can be finished in a finite time.

In contrast, ASICs have always been more complex and ASIC designers have suffered from long tool runtimes. Although performance features in current generations like multi-threading and load balancing across a cluster alleviated this problem it can still be beneficial to split the design into partitions and implement these partitions separately in parallel. It is paramount that these partitions have a clean interface to each other, the fewer signals between the partitions the better. Hundreds of signals do not only diminish the readability but will also worsen the chances of getting fast timing closure during final chip assembly. In order to benefit from time reductions of a parallel implementation each block should contain a similar amount of logic or, if this is not possible, at least a significant part.

Thus, the EXTOLL ASIC design is split into four partitions that lie in the same hierarchy level as seen in the following figure 2.25:

- **ht3\_partition**  
This module contains the HyperTransport PHY, the corresponding control logic and the HyperTransport 3 core. The three virtual channels define the interface to the adjacent block.
- **pcie\_partition**  
Similar to the HT3 partition this block contains the PCIe host interface logic

consisting of the PCI Express PHYs, the PCIe Gen3 core and the PCIe bridging logic. The interface to the next partition is also defined through the virtual channels.

- **extoll\_partition**

This partition connects to the two host interfaces and contains the core EXTOLL logic, including the HTAX bridge, the HTAX crossbar, the different functional units as well as the network ports. The network port interface is used to connect to the last partition.

- **extoll\_network\_partition**

The last partition contains all network related modules. The central part is the EXTOLL crossbar which connects to the link ports. The serializer control logic as well as the serializers is also placed in this partition.

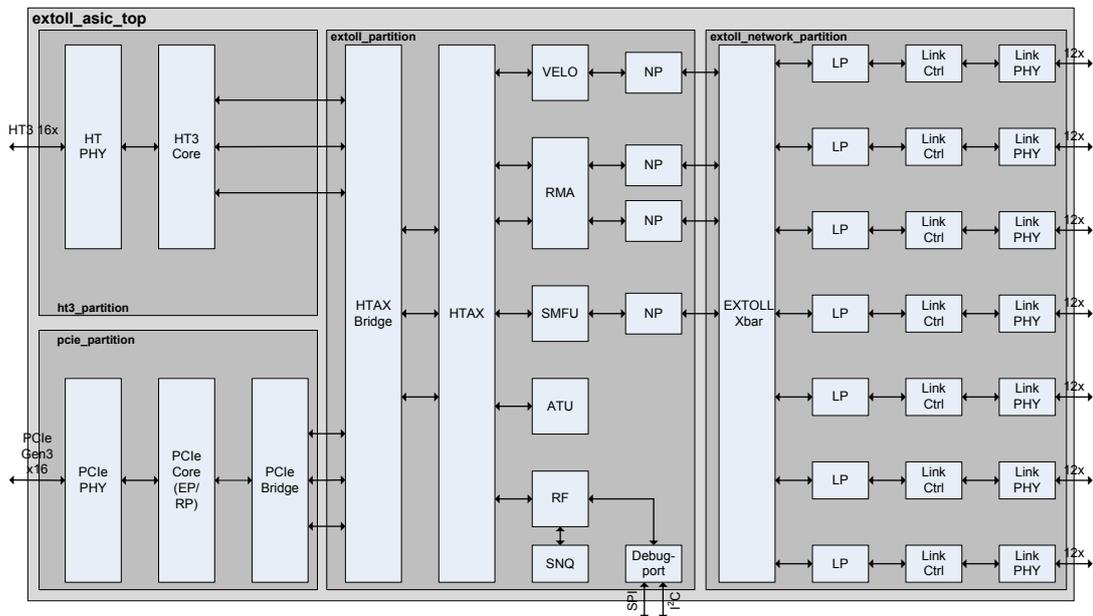


Figure 2.25 ASIC Toplevel Organization

Besides these four major modules the EXTOLL ASIC toplevel design still contains several minor modules that take care of the clocking in the chip as well as reset management and the control of the patch engine described in paragraph 2.5. The JTAG macro and BIST control logic is also placed at the top of the design hierarchy.

While the serializer macros for both the link and the PCIe interface are embedded in their respective partitions all CMOS I/O cells are clustered in the toplevel.

In order to easily distinguish physical nets (i.e. nets that are connected to pads) from logical nets the following naming convention was introduced:

- All external connections are identified by uppercase letters and routed to the corresponding I/O cell or hard macro.

All internal connections coming from the I/O cells or macros are labeled in lowercase letters.

Although the number of signals between the partitions should be kept as small as possible there are still numerous wires that must be connected. Fortunately, most of these connections are grouped together in logical clusters and the SystemVerilog HDL offers a construct to bundle signals in a so called *interface*.

The following code example shows an interface definition for the connection between two hierarchies in the register file.

```
interface rf_interface #(parameter AWIDTH=8, parameter TWIDTH=1,
parameter RWIDTH=64, parameter WWIDTH=64) ();
  logic [AWIDTH-1:0] address;
  logic [RWIDTH-1:0] read_data;
  logic                invalid_address;
  logic                read_en;
  logic                write_en;
  logic [WWIDTH-1:0] write_data;
  logic                access_complete;
  logic [TWIDTH-1:0] triggers;

  // ...

endinterface
```

By default the signals in the interface do not have any direction information. Access to the interface signals, however, can be limited by including another SystemVerilog feature, the *modport*.

```
// interface definition

modport master (
  output address,
  input  read_data,
  input  invalid_address,
  output read_en,
  output write_en,
  output write_data,
  input  access_complete,
  input  triggers
);

endinterface
```

As seen above the modport is included in the interface section and enhances the signals from the interface definition with direction information. Typically the modport name denotes the role of that interface instance, in most cases there will be two modports in an interface that define how data flows through the interface, e.g. master/slave, sender/receiver or a similar denomination.

In the toplevel an interface instance is created and connected to the interface port of the respective partitions as seen in the following code snippet. In this example the number of ports that have to be connected is reduced from 8 to 1 for a single register file interface.

```
rf_interface #(
    .AWIDTH(`RFS_PCIE_RF_AWIDTH),
    .RWIDTH(`RFS_PCIE_RF_RWIDTH),
    .WWIDTH(`RFS_PCIE_RF_WWIDTH)) pcie_rf_if ();

pcie_partition pcie_partition_I (
    // other signals
    .pcie_rf_if ( pcie_rf_if )
);

extoll_partition extoll_partition_I (
    // other signals
    .pcie_rf_if ( pcie_rf_if )
);
```

The use of the interface constructs adds another abstraction layer to the design and moves the design process in the direction of a system-level design where the actual implementation details are hidden in a lower hierarchy level.

Reusability is also improved: in the case of an interface change like an additional signal it is only necessary to change the interface definition and the change is reflected in every interface instance. Thus, the change is only visible to the two endpoints of the interface. All layers in between are signal agnostic. Without the use of interfaces additional wires would have to be connected throughout the complete hierarchy causing lots of work for the designer.

Unfortunately, current FPGA design tools do not completely support interfaces so that they were only used for connecting the toplevel in the ASIC design and all other modules still use the old port syntax.



## 3 ASIC Design Considerations

ASIC design is a highly complex process. In order to succeed with a working implementation it is not only necessary that designers follow the guidelines presented in the previous chapter to prepare a data basis that will lead to as few problems as possible but all the other steps from RTL to tapeout must be planned and executed carefully. High complexity of today's multi-million-gate designs lead to tremendous tool runtimes. Therefore it is necessary that design teams are able to work concurrently on the same design to reduce turnaround times if parameters in the design flow must be adjusted.

The following chapter outlines some challenges in the ASIC flow and shows a methodology to develop an efficient floorplan. Due to the close collaboration of the research group with Cadence Design Systems [57], one of the major EDA companies in the world, the flow is implemented with the tool chain from Cadence, namely *RTL Compiler* [58] for synthesis and *Encounter Digital Implementation* [59] for the backend flow as well as surrounding support tools.

### 3.1 Design Preparation

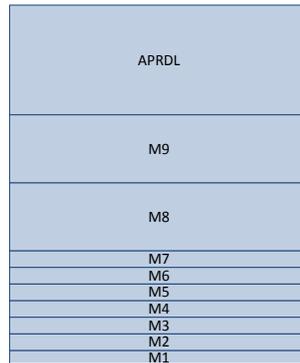
For the complete design flow a set of libraries and technology files is needed and should be prepared or obtained in advance. The following subsections will give a short overview of the required files throughout the design flow.

#### 3.1.1 LIB

The Liberty file format [60] was introduced by Synopsys in 1987 and subsequently provided as open source in 2000. It has become the industry standard for timing libraries and is supported by virtually every EDA tool. It contains information about the logical function performed by the cell, the cell area, pin capacitances, power consumption and dissipation as well as timing arcs for the propagation delay through the cell depending on the input value. The format has been enhanced over the time to incorporate information that is needed to model the effects of Deep Sub Micron (DSM) technologies more accurately. Liberty files also include wire load models to estimate the net delay between cells and are usually supplied by the IP provider. For proprietary IP .LIB files must be generated using Spice simulations.

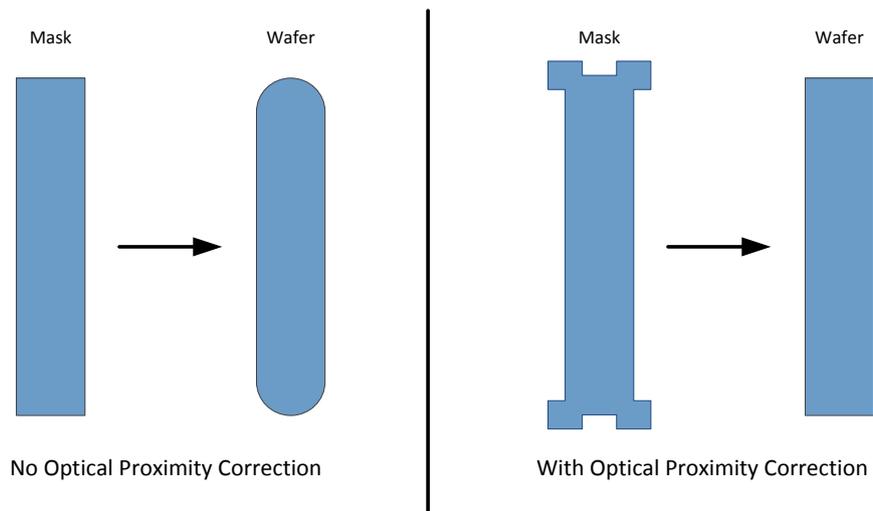
### 3.1.2 LEF

The Library Exchange Format (LEF) [61] was introduced by Cadence Design Systems and provides physical layout information for backend tools. Usually, the foundry provides a technology LEF that contains information about the layer stackup, the vias that are allowed for the technology and design rules for spacing the tool must adhere.



**Figure 3.1 Layer Stack of EXTOLL ASIC**

The format has evolved since its introduction to allow the definition of more and more design rules that must be followed to avoid chip defects. Due to lithographic effects the structures on the masks do not represent the structures that appear on the wafer. In order to get the desired results on the wafer the masks must be altered by so called Optical Proximity Correction (OPC).



**Figure 3.2 Effect of Optical Proximity Correction**

As seen in figure 3.2 the wire on the right side must be extended with notches on the mask so that a rectangular shape appears on the wafer. The LEF must include rules (so called end-of-line rules) that take these notches that are added after tapeout into account so that no spacing violations and unwanted shorts occur. The smaller the technology nodes have become the more design rules had to be added to the LEF.

LEF files also provide an abstract view for cells and hard macros in the design. These LEF files are supplied by the IP provider or must be generated from the analog layout tool for proprietary cells. Since the Place&Route tool does not need the full layout information of the transistors inside the cell it is sufficient to provide the size of the cell and the location and shape of the cell's pads as shown in the following exemplary inverter in figure 3.3.

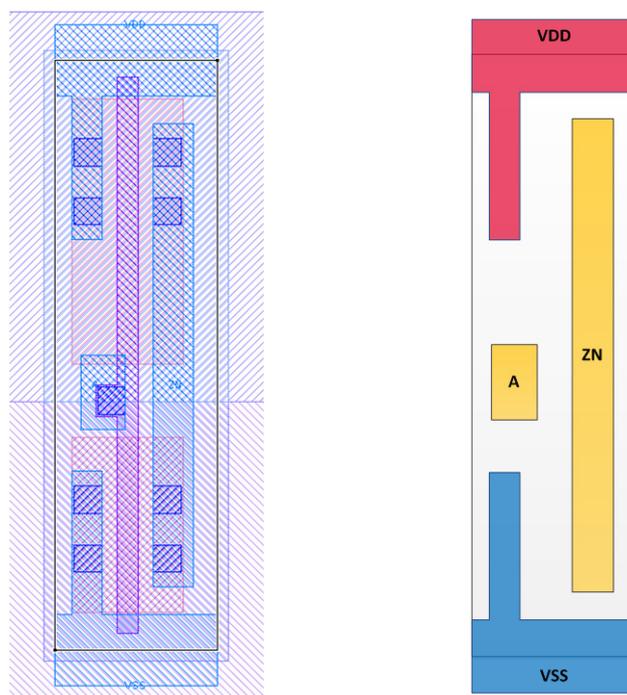


Figure 3.3 Simple Inverter as Full Layout and Abstract View

### 3.1.3 Capacitance Tables

Although the LEF file contains parasitic information to model wire delays it is a generalization that was created by the foundry for a specific process. For Deep Sub Micron designs a more accurate definition is needed because of the variations that happen for different process corners. A capacitance table is generated from an Interconnect Technology (ICT) file that is provided by the foundry and contains detailed information about the

semiconductor process (e.g. layer thickness, dielectric constants, material composition etc.) that can be read by a specialized field solver that calculates a capacitance table from these technology parameters. This process must be done for all available process corners so that the information can be taken into account during the design flow.

### 3.1.4 SDC / Timing Constraints

The Synopsys Design Constraint (SDC) format [62] is based on the TCL language and used to specify timing information for a design. It was originally developed by Synopsys but has been adopted as standard by all major EDA vendors. In order to run timing analysis and perform optimizations the tool has to know about all the clocks in the design. The following code defines the main EXTOLL clock with a frequency of ~770 MHz and relatively large margins for setup and hold analysis. During early design stages when no information is available about the physical implementation of the clock routing the constraints are usually more imprecise (and thus harder to achieve) but they will become more accurate later in the design process.

```
create_clock -name extoll_clk -period 1.3 \  
[get_pins extoll_clocking_I/extoll_clk]  
  
set_clock_uncertainty 0.30 -setup extoll_clk  
set_clock_uncertainty 0.05 -hold extoll_clk
```

If clock dividers as described in paragraph 2.6.1 are used the tool has to know the relation between the parent clock and the derived clock:

```
create_generated_clock -name extoll_clk_by_5 -source clkdiv5/CK \  
-divide_by 5 -duty_cycle 50 clkdiv5/CK_OUT
```

Based on the delay of the digital circuit that performs the division timing analysis can also calculate the phase relation between the two clocks. Another important information is the definition of paths that must not be analyzed. Clock domain crossings between two independent clocks cannot be analyzed because the tool is not able to define a relationship between them. Therefore, these *false* paths must be excluded from timing analysis. It is the designer's responsibility to take care that these paths are correctly handled by using suited synchronizing circuits to avoid metastability.

```
set_false_path -from [get_clocks extoll_clk] \  
-to [get_clocks pcie_clk]  
  
set_false_path -to [get_clocks extoll_clk] \  
-from [get_clocks pcie_clk]
```

At last timing relaxations can be defined for certain paths where the designer knows that it is not important whether a signal arrives one or several clock cycles later. In these cases defining multi-cycle paths can help achieving timing closure.

```
set_multicycle_path 2 -setup -through ram_wrapper/pins_out/sec
set_multicycle_path 2 -setup -through ram_wrapper/pins_out/ded
```

Usually, a design does not consist of only one SDC file. Timing constraints can be defined for different use cases, e.g. the test clock does not have to be constrained for normal operation. The tools, however, have to optimize and analyze for all possible use cases.

### 3.1.5 DEF

A Design Exchange Format (DEF) file [61] contains the current state of a design in the design process and can include both logical design data like constraints and physical design data like placement locations and routing information. Together with the information in the LEF files it is a complete representation of a chip's design data. Its primary purpose is the data transfer between different tools (design databases are usually stored in a proprietary format). A DEF file does not have to include every possible feature but can be limited to store only a subset of them, e.g. for transferring a floorplan from backend to frontend for physical synthesis the DEF does only have to include the floorplan information.

### 3.1.6 Flow Script

Most EDA tools nowadays can be controlled through a graphical user interface. However, many commands and switches cannot be reached via the GUI. Full control of the tool and the flow can only be gained by using the integrated TCL based command interface. This does not only allow the execution of a complete flow through a set of interwoven scripts by running commands sequentially but also more complex design manipulations by accessing the design databases through TCL commands and performing batch operations on them.

For example, in Encounter a bump of a flip chip die can only be assigned to an I/O driver pin. However, this I/O pin can be anywhere in the hierarchy and might also change in the design process. It is much easier to assign a net name to a bump. The following TCL procedure introduces a function that takes a bump and a net name as parameters, searches the design database for the cell that drives this net and connects the drivers output pin with the associated bump:

```
proc assignBumpByNet { bump net } {
  set iobuffer \
    [dbGet [dbGet -p top.nets.name $net].instTerms.inst.name]
  set iopin [string map [list $iobuffer/ "" ] \
    [dbGet [dbGet -p top.nets.name $net].instTerms.name]]
  assignIOPinToBump -buffer $iobuffer -pins $iopin -bump $bump
}
```

If the tools supported only GUI access neither such enhancements nor a simply reproducible flow could be achieved in chip design.

### 3.2 Data Hierarchy

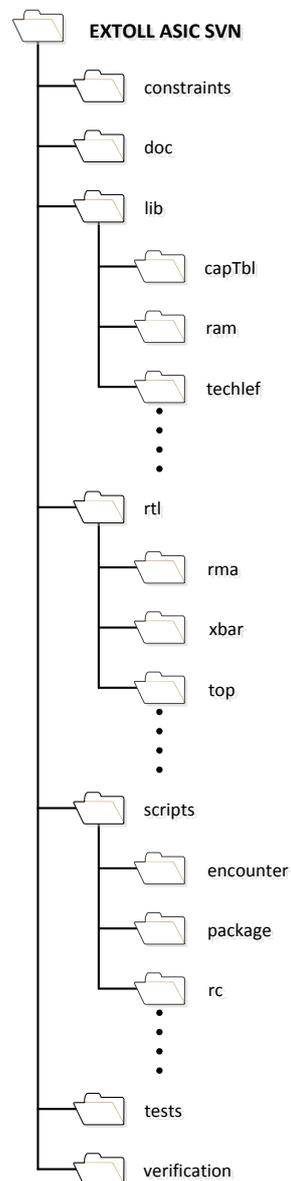
Chip implementation tools have become more and more integrated over the last few years. While it was common to run separate tools for different steps in the design nowadays the whole subset of functionality can be controlled from a single tool even though the legacy tool might still run in the background. Nevertheless, the user does not have to setup each tool separately and can work in a common environment with a single setup.

The required input for the synthesis can be divided into technology files and user-generated content. Technology files like libraries for standard cells or hard IP stay the same across several projects at the same process node and should be stored at a central location on a network drive to be easily accessible. Project dependent content like RTL, constraints or TCL scripts to control the tool flow is usually stored in text files and can change over the time. Storing this data in a version control system like *subversion* (SVN) [63] keeps track of changes made during the project and also allows other designers to work on the files at the same time. However, project dependent libraries like customized memories can also be managed through the version control system. Figure 3.4 shows the directory structure of the SVN that is used for the EXTOLL ASIC. All the design data is stored in a *rtl* folder while the flow scripts are stored in a separate *scripts* folder. Other folders contain documentation files (*doc*), constraint files (*constraints*), project libraries (*lib*) and verification related code (*tests*, *verification*).

Designers can checkout the complete design database and its associated control scripts and run the complete frontend and backend flow in their local working directory anywhere in the network.

The results from the different steps are handled in a similar manner. While running localized flows generates all the intermediate databases in the user's directory structure known

“golden” databases are stored at a central network location so that the designer can enter at any point in the flow to optimize some details without having to rerun everything, a process that could take several days for a complex design like EXTOLL.



**Figure 3.4 EXTOLL ASIC Subversion Structure**

The distributed organization of the design flow with centralized storage for important information also allows several designers to work on the same project concurrently. They can modify design scripts at the same time because the version management system can merge the changes made by different users. Since intermediate results are stored centrally, one

## ASIC Design Considerations

---

designer can review a specific step in the design flow while another reviews the step afterwards.

The following table 3.1 summarizes the types of contents and their storage locations.

	Content	Storage
Input data	EDA libraries (static content)	Central network location
	RTL, flow scripts etc. (dynamic content)	Version management
Output data	“Golden” results	Central network location
	Intermediate user results	User specific location

**Table 3.1 Design Data Storage**

Despite these techniques to ease collaboration ASIC design is a rather linear flow which will be presented in the following paragraphs. The flow does not allow for much parallelization of the actual design tasks since most scripting and adaptations can only be done after the previous step has been completed successfully since the results rely on each other. However, there are always some side tasks that can be carried out in parallel because they are not directly related to the main flow (equivalence checking, for example).

A group of not more than four people can carry out the complete design flow efficiently without much idle time in between; more resources assigned to the task will not improve the efficiency any more.

### 3.3 Frontend Flow

In early days libraries for EDA tools were strictly separated in front-end and back-end views. Usually, front-end design teams and back-end designers had little contact and worked separately (so called throw-over-the-wall approach). Front-end designers required little to no knowledge of the backend flow and vice versa. Synthesis tools only required timing libraries to determine cell delays. Wire delays between standard cells were analyzed by using statistical models, e.g. for a block of  $n$  gates a driver with a fanout of  $m$  is usually connected to a wire with a delay of  $x$  ps. This worked well for designs on large process nodes where cell delay was significantly higher than the wire delay. In DSM nodes the reverse is true, the total propagation delay of a path is mainly determined by its wire delay [64]. The crossover point has been reached at the 0.13 $\mu$ m node as shown in the graph in the following figure 3.5.

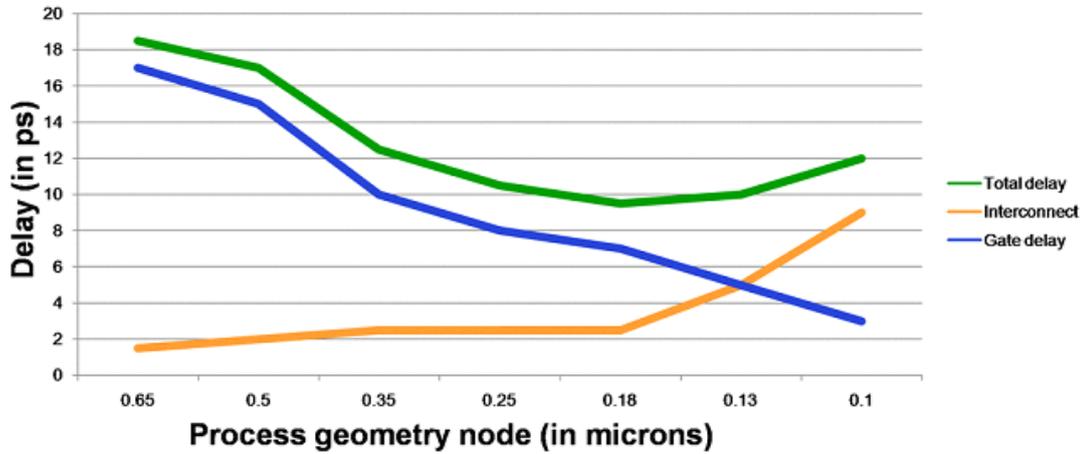


Figure 3.5 Gate vs. Interconnect Delay<sup>4</sup>

As a first step it is important that the synthesis tool already has information about the size of the cells and the intended technology to model wires with more accurate delays. The following figure 3.6 summarizes these requisites.

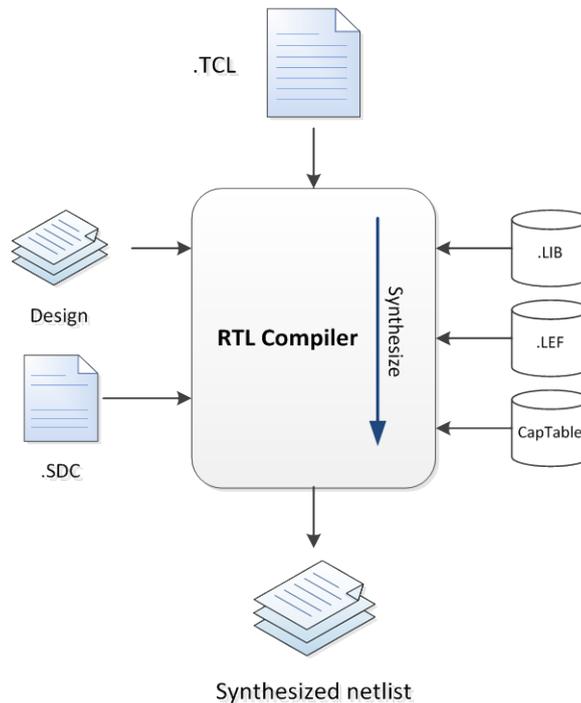


Figure 3.6 Synthesis Input and Output Files

<sup>4</sup> Courtesy of IDESA, an European Community FP7 project, [www.idesa-training.org](http://www.idesa-training.org)

Knowledge about the technology allows the tool to run a prototype placement without constraints during synthesis that is able so that a first estimation of the final wire length can be achieved. For better correlation between frontend and backend the floorplan (either preliminary or final) can be back-annotated to synthesis so that the placement estimation and thus the resulting wire delays become more accurate.

This also allows the synthesis tool to direct its optimization efforts to paths that actually need to be optimized. With a statistical model it is possible that the tool wastes computing power on paths that will already meet timing because the gates in it are placed much closer than assumed by the model. Because of the way how a synthesis tool works this can become a problem for cases with many paths that are optimized unnecessarily. A synthesis tool usually has a time budget allocated for optimization efforts so that it does not run indefinitely in cases of unreachable timing. When it has to optimize lots of unnecessary paths it can happen that it will stop its optimization efforts sometime in the flow without giving attention to paths that are in earnest need of improvement.

The synthesis flow is a simple process that is shown in the following figure 3.7. Each step in the flow is associated with one or more TCL script that performs the tasks required for each step. In order to run the complete synthesis implementation only the master script *extoll\_rc\_flow.tcl* has to be sourced by the user.

In the following each step in the design flow is shortly introduced:

- **Load Libraries**

As previously mentioned synthesis nowadays requires a whole set of libraries and technology information. This information must be setup firsthand in the tool. Sometimes there are inconsistencies between the frontend and backend views that must be analyzed to let the tool know the more accurate definition.

```
set_attribute library [list required libs]  
set_attribute lef_library [list required LEF]  
set_attribute cap_table_file rcworst.capTbl
```

- **Load Design**

This step loads all the HDL files and elaborates the complete design. If any modules are not available these instances will be marked as black boxes and synthesis will continue without them.

```
read_hdl -define ASIC -sv [list all HDL files]  
elaborate
```

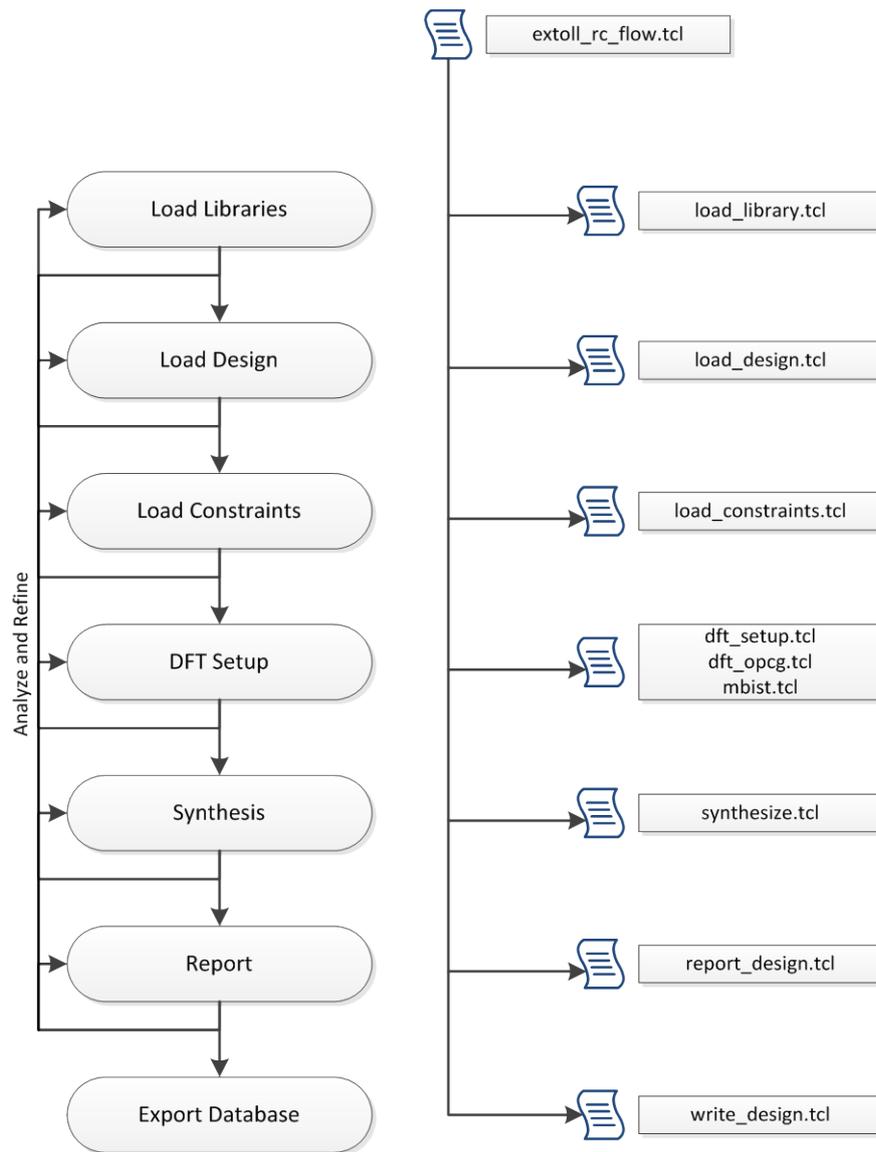


Figure 3.7 Frontend Flow Organization

- **Load Constraints**

Timing constraints have to be added to the design so that the synthesis tool has information about its optimization goals. If no timing constraints are defined the design is simply synthesized and mapped to the target technology, however, little to no optimizations are performed. Typically, a design can operate in different modes (e.g. functional mode for normal operation and test mode during test). Each of these modes requires a separate set of timing constraints. The tool will analyze and optimize timing for all defined modes concurrently. It is important that the

constraints are not too tight so that amount of time used on optimization is still reasonable.

```
create_mode -name {functional test }
read_sdc -mode functional functional.sdc common.sdc
read_sdc -mode test test.sdc common.sdc
```

At last, the back-annotated floorplan is also read in this design step to perform Physical Layout Estimation (PLE).

```
read_def floorplan.def
```

### - DFT Setup

While the previous steps only handled the loading of different design structures the DFT steps take care of lots of design modifications that are required for testing. First of all, basic DFT features have to be setup like the number of scan chains in the design and their type (compressed/uncompressed), which pins are defined as scan inputs and outputs and whether they are dedicated or shared with normal I/O pins and so on. The JTAG macro including its connections also has to be defined.

The RAMs used in EXTOLL are equipped with internal test structures and test ports. For normal operation and during development these ports are not used. Thus, the design instantiates wrappers that can utilize the test ports. However, the wrapper connections are tied off because the functionality is not needed during simulation. After the design is elaborated in the synthesis tool the RAMs are spread all over the hierarchy and the test signals must be connected to the test controller [65]. RTL Compiler allows extensive manipulations on the netlist so that all these connections can be made by TCL scripting.

The following figure 3.8 shows the process based on a simplified example.

At first all memories in the design are located and put in a list. In the first steps of the design manipulation the tie-offs are disconnected from all memories.

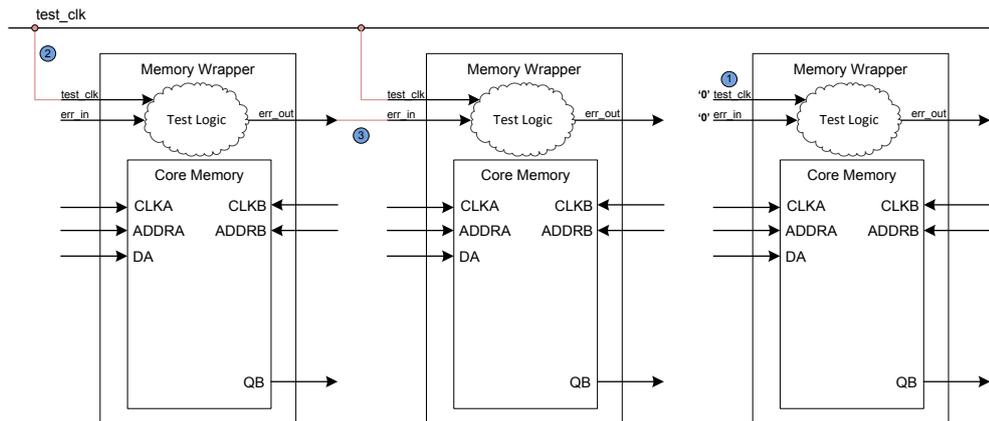
```
edit_netlist disconnect mem_wrapper/pins_in/test_clk
```

The second step ties all signals together that are driven by a single global signal like the test clock in this example.

```
edit_netlist hier_connect test_clk_root/pins_out/test_clk
mem_wrapper/pins_in/test_clk -in_prefix test_clk_in -out_prefix
test_clk_out
```

At last in step 3 the connections between the different instances are established like the shift register that holds the information about occurred errors in this example.

```
edit_netlist hier_connect mem_wrapper/pins_out/err_out
next_mem_wrapper/pins_in/err_in -in_prefix err_in -out_prefix
err_out
```



**Figure 3.8 Memory Test Connections**

Finally, logic is added around the PLL to allow on product clock generation during test.

- **Synthesis**

Synthesis is actually the step that requires only minimal input from the designer. Synthesis is performed in two separate steps. At first a general synthesis is performed that maps the high-level HDL to simple generic gates. In the second step the mapping to the target technology is performed together with optimizations that become possible due to special complex gates that are available in the technology. In between the two steps DFT violations that have occurred during synthesis must be identified and fixed.

- **Report**

In order to determine the quality of the synthesis, the achieved timing and area a whole set of reports is generated after synthesis concludes. These reports help to identify problems that might have occurred at some time in the flow.

- **Export Database**

After the synthesis flow has completed the design must be written out. RTL Compiler can directly write out a design database for Encounter that already includes the complete setup of the libraries so that these steps can be skipped in the

following backend flow. Besides the export to backend the synthesized netlist is also written as standalone Verilog file for simulation. Further verification can be performed by writing out setups for equivalence checking and constraint verification tools.

### 3.4 Backend Flow

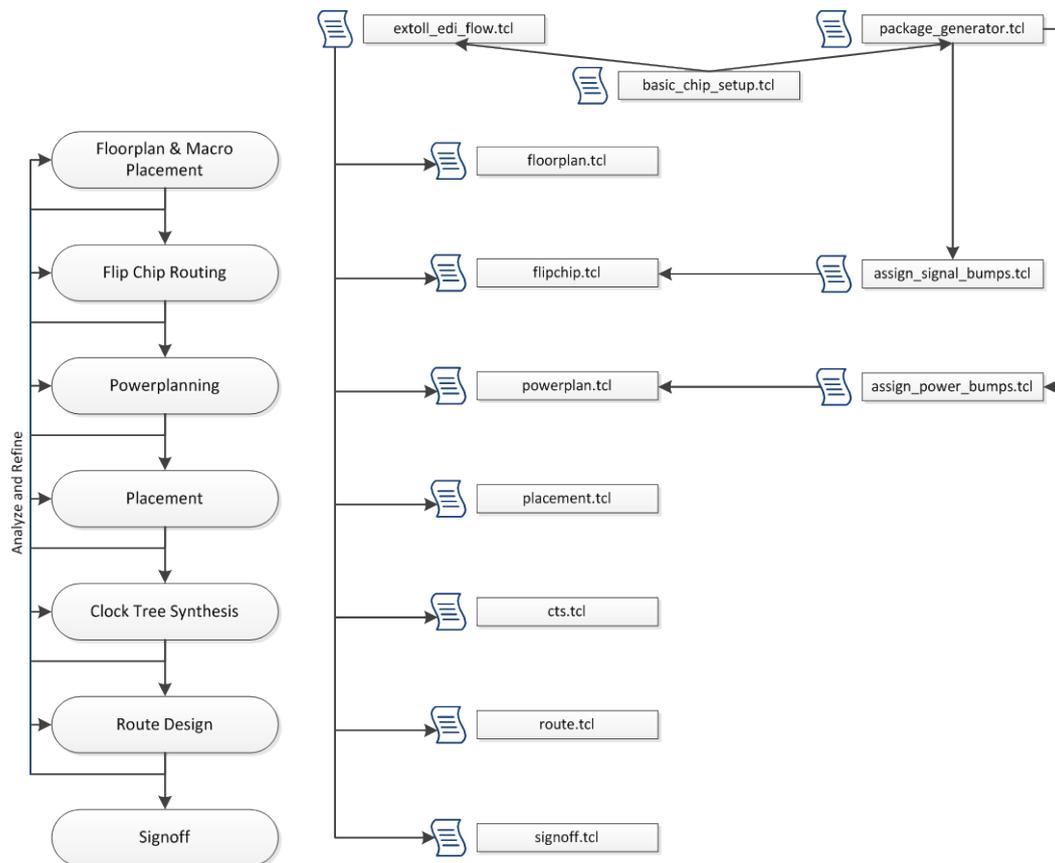
After synthesis is completed the design database must be imported by the backend tool. This is the only real transition of databases in the whole ASIC flow. Fortunately, the tools are designed to interoperate well so that RTL Compiler can already write out a complete environment setup for Encounter that is not only able to load the synthesized netlist but since all the physical design information in the libraries is already needed in the frontend also all the required libraries are read in correctly.

The backend flow consists of a set of tasks that are executed sequentially in the design process. Because a lot more process steps have to happen in backend the design flow is much more complex than the synthesis flow. Like the frontend flow the backend flow is script based. Each major step is controlled by a single TCL script file which can call additional scripts to split tasks further in cases where the design task becomes too complex to be handled by a single script. Although it would generally be possible to put every task in a smaller number of scripts this reduces both readability and the possibility to easily reuse scripts in upcoming designs.

Basic design information like the die size, the bump pitch etc. is outsourced to a common setup script *basic\_chip\_setup.tcl* that is shared with the scripts that are used for package generation (see paragraph 4.3). This means that as soon as one basic parameter has to be changed due to manufacturing reasons the information is automatically propagated to all relevant scripts and can be used for automatic regeneration.

```
set die_size_x 14000
set die_size_y 11600
set bump_pitch 200
set bump_edge_space_min 150
set bump_count_x [expr ($die_size_x - \
                      2 * $bump_edge_space_min)/$bump_pitch]
```

The following figure 3.9 shows the major steps that have to be executed as well as their associated script files.



**Figure 3.9 Backend Flow Organization**

In the following each step of the backend flow is introduced before the next paragraph 3.5 outlines the challenges during floorplanning.

- **Floorplan & Macro Placement**

Floorplanning is one of the most important tasks in the physical design process as it defines guidelines for the location of the cells in the netlist on the die. The process is paramount for reaching timing closure and usually requires several iterations. The process itself will be discussed in more detail in the following paragraph.

- **Flip Chip Routing**

In this step the bump array is defined and created. In the beginning all the bumps are just dummy bumps with no net connections. These connections must then be created logically by associating a driver pin with the respective bump location. In this flow this is done by executing a script generated by the package generator that annotates this information to the physical design. After everything is setup the specialized flip chip router routes the wires from the bumps to the drivers while

considering the constraints that are defined for the nets (e.g. differential routing to balance  $p$  and  $n$  or shielded routing for sensitive wires).

- **Powerplanning**

A correct power distribution is fundamental for correct operation of the chip. Power planning consists of several steps. At first, all cells in the design must be connected to power nets because no power pin is connected after synthesis. As soon as logical power connectivity is established the power grid that lies over the chip in the upper metal layers must be defined. After placement the power stripes of the standard cells will have to be connected to this grid. The bumps that deliver the power from an external voltage regulator must also be connected to the power grid which is done by back-annotating the information from the package. At last, power analysis must be run carefully to simulate the expected IR drop due to switching activity all over the chip. If the voltage drop exceeds the specified boundaries the power grid must be redesigned to meet the new requirements or additional decoupling cells have to be added to the affected region.

- **Placement**

The Placement step takes care of placing the cells on the die. For this purpose it adheres to the guidelines defined during floorplanning and also considers all the cells that were placed in a previous design step.

- **Clock Tree Synthesis (CTS)**

In this step the clock trees for the design are built. In order to do this the tool requires input about all the clocks in the design. The user also has to define the cells that are allowed to build the tree. In the synthesized netlist clock connections are simple direct connections between the clock root and the clock pins at the gates. However, since the clock is a high fanout net with a corresponding capacitive load it is organized in a tree structure (hence the name clock tree) with buffers in between to distribute the load across the tree. It is the task of the CTS to build a tree that is balanced. A tree can be built either with buffers or inverters (two successive inverters perform no logical operation), however, the cells must be symmetric, i.e. the rise time and the fall time must be identical. The respective cells are logically added and physically placed into the design by CTS. Afterwards the clock tree is routed as long as the normal wires are not yet routed since the logic is sensitive to clock skew, especially in a high performance design with almost no timing margin. Clock nets can also be routed with shielding to reduce the possibility of cross-coupling or other external influences. Besides clock nets CTS also performs routing of the reset net in a

similar way since it has the same requirements for balanced routing as the clock nets and is also a high fanout net.

- **Route Design**

In this step the remaining connections in the design are routed which means millions of wires in a complex design like EXTOLL. During the routing process the router tries to incorporate timing constraints (e.g. paths with a small timing budget must be routed as short as possible) and signal integrity considerations (e.g. long parallel routes facilitate crosstalk). Because the process is computationally intensive it is split in two parts. During global routing the design is partitioned into cells and the wires are only routed between the cells. This already gives an insight if an area will be congested or not. In the following step, detailed routing the actual connections between the globally routed wires and the pins are created. There can be cases that some pins cannot be accessed or that the router creates shorts to connect a pin. Although the tool tries to fix these problems by ripping up nets in these areas and rerouting them it is possible that some violations will not be fixed automatically. In this case the designer must find a solution.

- **Signoff**

The final step before tapeout is the Signoff process. In this step the completely routed design is handed off to the signoff tool that will run a detailed 3D parasitic extraction to model all physical conditions. This is the most accurate analysis that can be done for the design, however, it can also take many days to complete. After timing analysis and power analysis based on the extracted model are completed physical layout verification is performed to check the physical design rules (DRC) and layout vs. schematic (LVS) that checks if the layout implementation that is extracted from the tapeout information matches the schematic representation of the design.

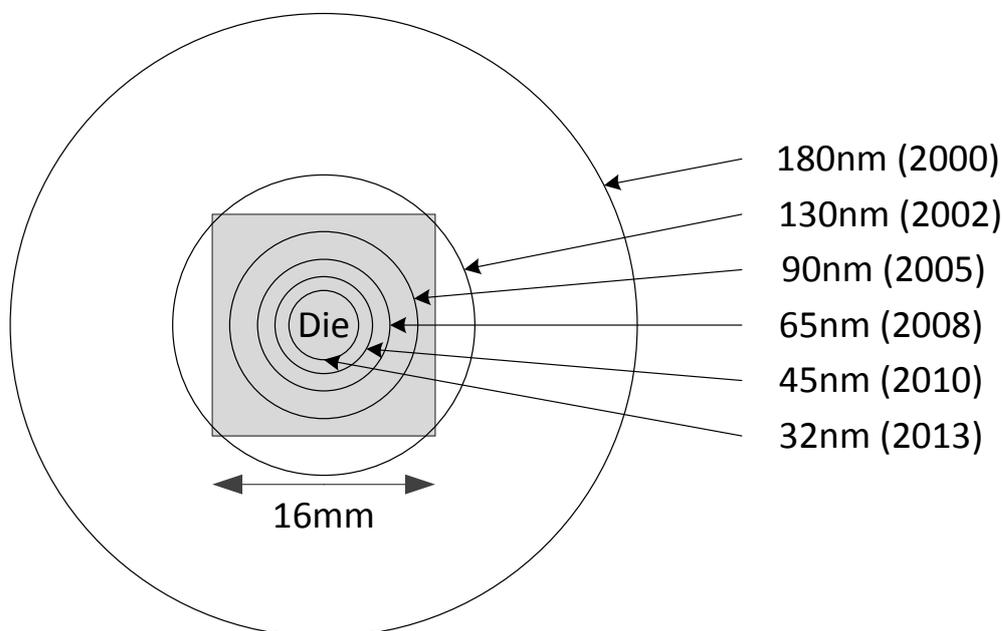
In between all these steps timing optimizations occur that are refined after new physical structures have been added to the design. Additionally, as described in the previous paragraph 3.2 after each step the database is exported so that the process can be interrupted and continued almost anywhere in the flow.

### 3.5 Floorplanning and Datapath Analysis

As mentioned before with smaller technology nodes the wire delay has become significantly more important than the gate delay. This means that the placement of logic cells in relation to each other determines the maximum achievable clock frequency rather than the delay

caused by logic cells. For example, the propagation delay of an inverter in EXTOLL's 65nm process is well below 10ps, the target clock cycle time on the other hand is about 1.3ns. Modern SoC chips like EXTOLL utilize a large die area which means that logic cells can be spread apart more than one centimeter, a distance that cannot be crossed during a single clock cycle [66].

Steve Scott, Cray's former Chief Technology Officer presented the following figure in his keynote during the 15<sup>th</sup> International Conference on Parallel Architectures and Compilation Techniques (PACT) [67] that gives an excellent representation of the signal reach problem in modern SoC designs.

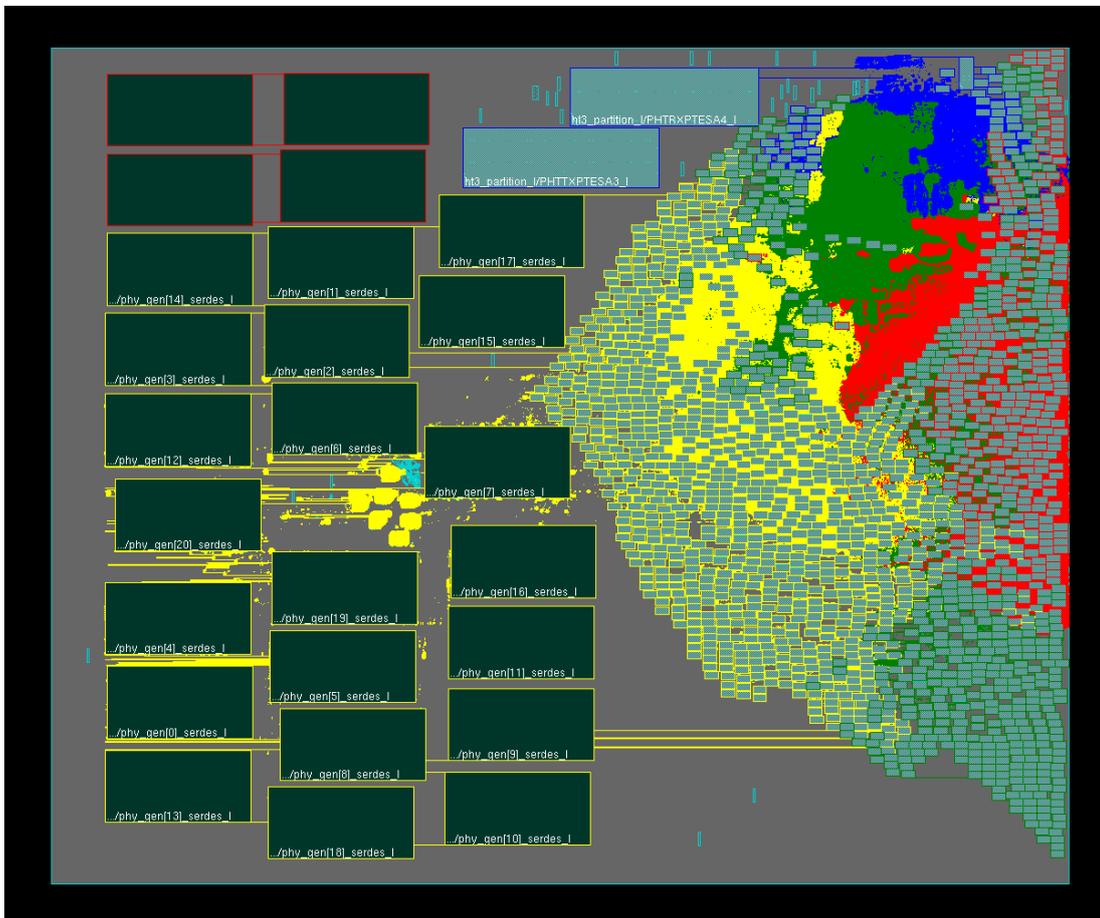


**Figure 3.10 Signal Reach in One Clock Cycle (8 FO4)**

As such Floorplanning is one of the most important steps in the backend design flow. It defines where the tool places the design instances. This has an effect on the routing implementation later on and the achievable timing.

While the backend tool is able to analyze connections between modules to cluster logic belonging to each other in nearby locations it is not able to determine the designer's intentions. Figure 3.11 shows an example of a placement that was automatically generated by the tool with no external constraints. Each of the four partitions defined in paragraph 2.8 have been colored (PCIe partition in red, HT partition in blue, EXTOLL partition in green and the EXTOLL network partition in yellow) to show how the tool placed related logic.

However, the result is far from optimal. First of all most of the logic is placed in one corner of the chip while the serializer blocks are placed in the other corner. This already leads to long wire delays for the paths that have to connect from the logic cells to the serializer macros.



**Figure 3.11 Automatic Placement without Floorplan**

Fortunately, the designer who has an idea how the logical structure will map to the physical implementation can support the tools with several placement guidelines that differ in their strictness:

- The Guide is the most relaxed constraint. As the name says it guides the tool where to place cells. The tool tries to adhere to the constraint by placing cells in the vicinity of the guide. However, a guide which can be set on modules or groups of modules in the logical hierarchy is a soft constraint and the tool is not forced to obey it.
- The Fence is the most restrictive floorplanning constraint. It has a fixed boundary that limits the module and all its children to the defined area. At the same time, logic

that does not belong to the sub-hierarchy of the module is not allowed to be placed inside this region. A Fence should only be used if it is necessary or if the designer is absolutely sure of his intentions. Since the tool is not allowed to override the constraint it can lead to suboptimal placements.

- The Region constraint is related to the Fence. Like the Fence constraint it forces a module or group with all its children into a fixed area. However, the boundary is not restrictive for the rest of the design so that other modules may overlap into the Region. This allows the tool to optimize placement in some areas where it thinks that moving instances into the Region might be beneficial to the timing.

A nice side-effect of constraining the location of modules is the fact that it greatly reduces the calculation time (placement is an NP-complete problem [68]) that the tool requires finding a global solution and can spend more time on optimizing placement within the modules. For a large design like EXTOLL placement can need several days until completion, a process that can be sped up to several hours by utilizing multi-threading functionality.

### 3.5.1 Datapath Analysis - Global View

In order to define guides for the placer it is important to analyze the dataflow in the chip which has direct impact on where modules should be placed. A block level schematic already gives a good overview how modules and functional units inside the design are connected. On a large scale it can be said that data packets traverse the chip from the host interface to the link interface and vice versa. With a closer look at the following figure 3.12 it can be seen that the dataflow itself is divided into two parts with the functional units being the transition point between the two packet types:

- Packets coming from the host interface can be routed to all the functional units and the other way round (denoted in light blue)
- The packets coming from the network interface can be routed to the functional units or other network links (denoted in light red)

Both packet types are distributed through a central crossbar across the chip. The HTAX crossbar switches host interface packets while the EXTOLL network crossbar switches the network packets. Because these crossbars interconnect all the modules that are connected to them they must spread over the chip in vertical direction while being close enough that timing closure can be achieved.

The block level schematic is already a good starting point for a first floorplan definition so that it can be mapped directly to the die. While it is relatively coarse grained it gives a good impression of the designer’s intention and dissuades the tool from placing modules at unreasonable locations. However, it does not show any information about minor datapaths in the design like the connection between the main register file and its children.

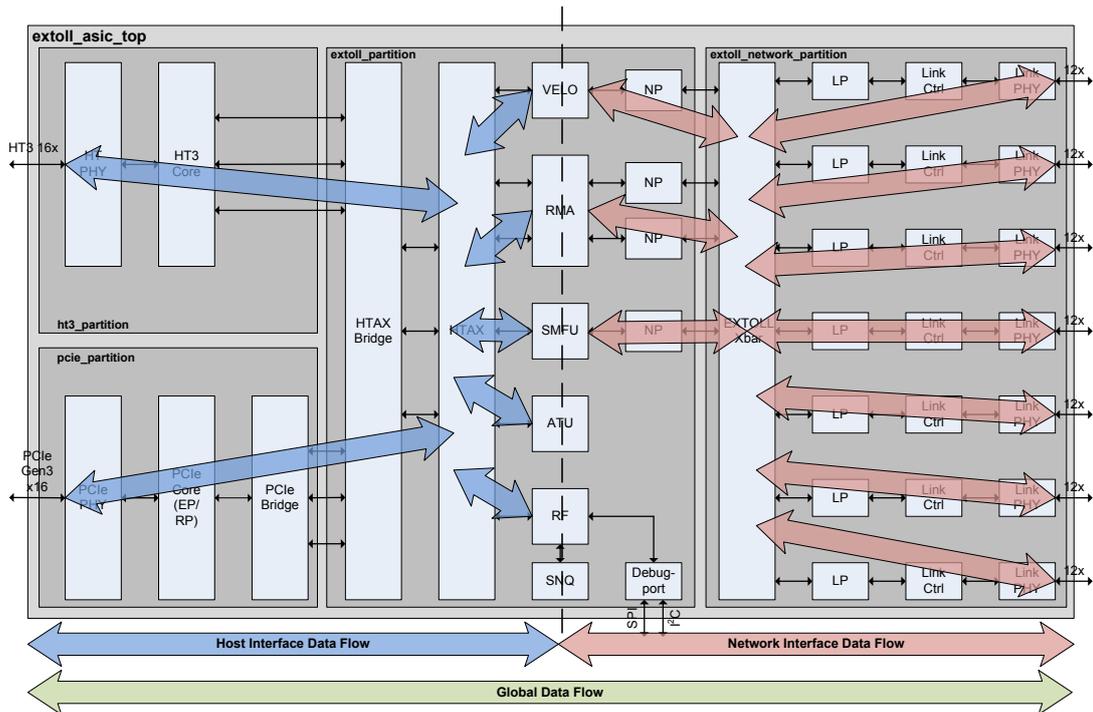


Figure 3.12 Global Datapath Analysis

### 3.5.2 Pre-placement

An article that appeared in EETimes in 2004 [69] stated that “hard macros will revolutionize SoC design”. It analyzed that the number of hard macros in a SoC grows exponentially and that the chip area is more and more determined by the area of these macros. Instead of a “sea of cells” which was common about a decade ago we now see a “sea of hard macros” which is an observation that can be confirmed by looking at figure 3.11.

The first improvement to the floorplan is the correct placement of the large serializer macros. Although the steps performed are an actual placement it is considered to be a part of the floorplanning process.

## ASIC Design Considerations

---

The location of the serializer macros is tied to the package development which will be discussed in the following chapter 4. Since the serializers cannot be placed at arbitrary locations they must be put at predefined positions so that their I/O pads align with the bump location defined in the package or at least in its vicinity.

The following figure 3.13 shows the result of an automatic placement after the serializer macros have been fixed at their correct location.

The resulting placement is already a large improvement to the result seen in the previous figure 3.11. The location of each partition already resembles the structure from the block level diagram (mirrored along the y-axis). Nevertheless further improvements can be made e.g. by defining module guides so that the free space between the links is utilized more efficiently.

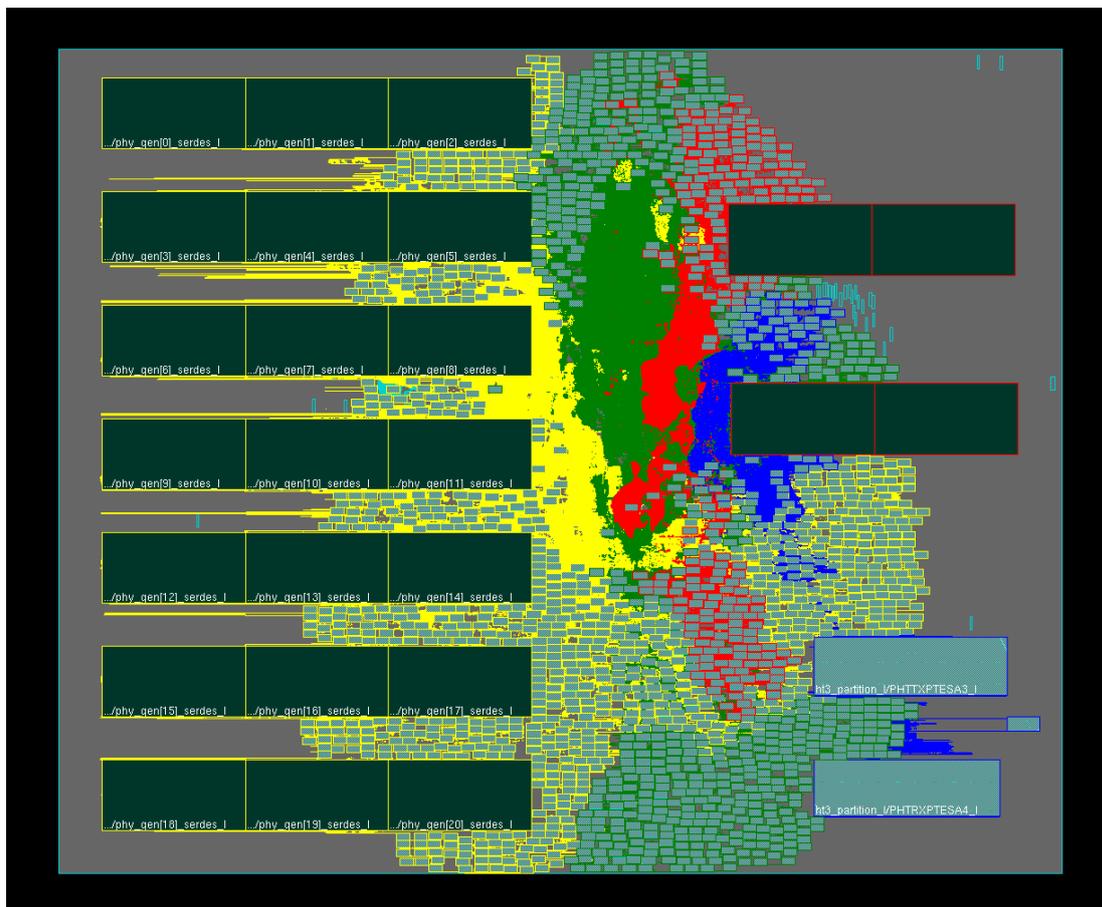
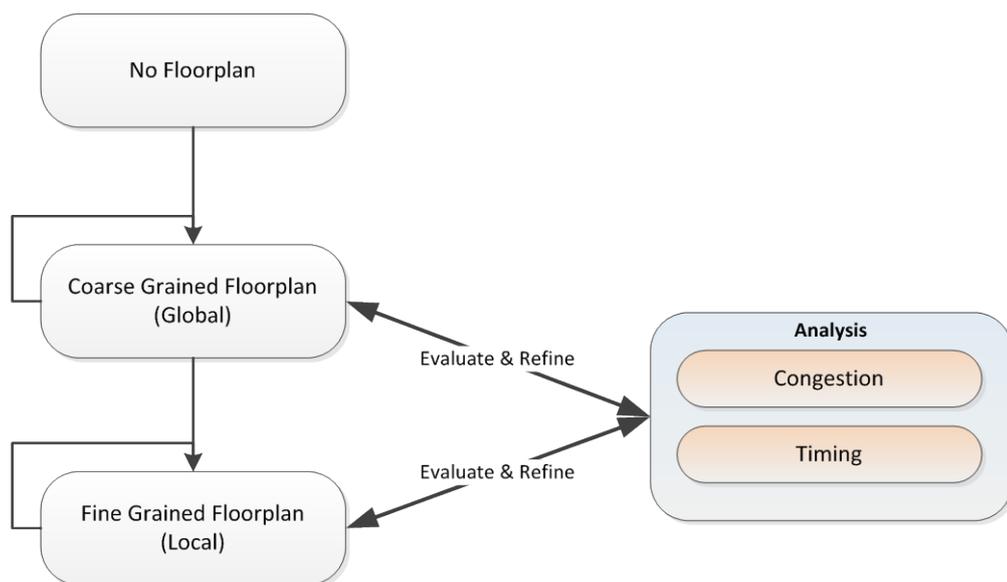


Figure 3.13 Automatic Placement with Preplaced Macros

Modern placers are suited to place objects of different heights (i.e. not standard cell heights) efficiently. Thus, it usually does not make sense to preplace RAM macros which could be a tiresome process considering that there are more than 1,000 instances in EXTOLL. However, situations can arise in the design process that might require preplacing at least some of them. Such detailed floorplanning operations are discussed in the following paragraph 3.5.3.

### 3.5.3 Datapath Analysis - Detailed View

Floorplanning is a hierarchical iterative process as shown in the following figure 3.14. It starts without any floorplan at all and continues with a coarse grained floorplan that maps the logical structure of the design to the die area. As soon as the global floorplan is satisfactory critical regions can be floorplanned in more detail. During and after each step the floorplan should be analyzed to see if it caused improvements and did not actually worsen the results.



**Figure 3.14 Hierarchical Floorplanning**

Analysis of the floorplan's quality can be conducted by looking at two parameters:

- **Congestion**

After the floorplan is defined a Trial Route process is called that performs a complete routing of the chip. However, for performance reasons it does not care about legal placement of the routes so that routing channels might be over-utilized. Nevertheless, it gives an indication if the design is routable at a later stage by looking

at the percentage of congested paths in the design. If one area is highly congested it might be a good candidate for further floorplanning to resolve issues that are caused by tightly packed logic.

- **Timing**

The result of the Trial Route process are typically direct Manhattan style [70] routes and therefore the shortest possible connection between two endpoints. This means that analyzing chip timing based on this routing will give the designer feedback on the quality of the floorplan. If the starting and end point of a path are placed far apart because they belong to different floorplan objects this will have a negative impact on the timing slack of the path. The sum of all negative (i.e. failing) paths will represent a measurable metric that can be used as comparison between several implementation runs.

The following table 3.2 shows the result of the first two floorplan iterations described in the previous paragraphs which show a tremendous improvement in the 2<sup>nd</sup> iteration..

Floorplan		No Floorplan		Macros Preplaced	
<b>Timing</b>	TNS (ns)	-16300000		-1750000	
	Violating Paths	702000		684000	
<b>Congestion</b>	Routing Direction	<i>Horizontal</i>	<i>Vertical</i>	<i>Horizontal</i>	<i>Vertical</i>
	Congested Paths	0.98% (658937)	3.57% (2413961)	0.01% (8466)	0.61% (407206)
	Uncongested Paths	99.02% (66913224)	96.43% (65168556)	99.99% (67563375)	99.39% (67175019)

**Table 3.2 Floorplan Evaluation**

While it might seem conducive to floorplan every module down to the lowest level, overconstraining can be just as harmful as underconstraining [71]. It is important to concentrate on modules that will benefit because of their structure (e.g. connections over large distances) or because of their logic complexity.

For the remaining modules it is most likely sufficient to floorplan only the upper design hierarchy. Only if the results that the tool produces are not satisfactory the designer should descend into submodules and optimize them separately.

Because of their logical structure three critical modules can be identified in the EXTOLL design that could benefit from a more accurate floorplanning:

- **Register File**

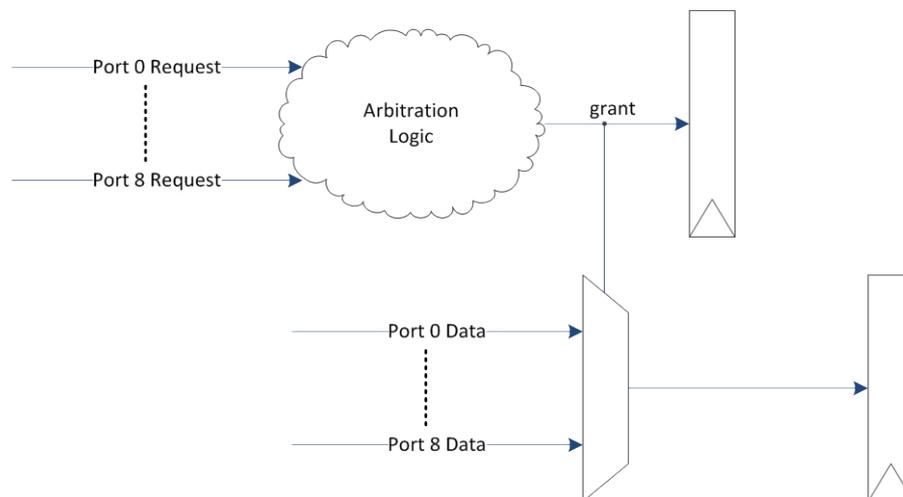
Each functional unit has its own register file instance that contains control registers as well as status and debug values. It became already apparent early in the design phase during FPGA prototyping that holding all information at a single location is detrimental to reaching timing closure. Therefore RFS already includes support for hierarchical decomposition by introducing external register files that are connected via a simple access interface to the main register file. Since read and write access to the register file from the host system is not latency critical the connection between the main register file and its children can be pipelined with one or more intermediate delay elements that simply introduce an additional register stage in the data and control path so that crossing large distances on the chip is not a problem any more. Furthermore, for most status signals it is not relevant if changes are delayed one or more clock cycles and that they might be visible to the module a little bit later. Therefore multicycle timing constraints can be applied to these signals which can also relax the timing requirements for the module where the register file is embedded.

That means that despite initial assessment accurate floorplanning for the register file is not needed because of its design structure and the resulting timing improvements.

Global datapath analysis in paragraph 3.5.1 showed that the two central crossbars are also candidates for detailed floorplanning. Crossbars are by design limited in their scalability because the internal datapaths have to be muxed from all the inputs. Additionally, arbitration requires request signals from all inports. Of course, the crossbar should be as fast as possible. Therefore it usually is latency optimized which means that it will consist of as few register stages as possible. While this is a good aspect for performance, it is equally bad for layout on the chip, especially for high radix switching architectures [72].

- **HTAX**

The HTAX network-on-chip (NoC) architecture is highly optimized for latency. Its internal structure is shown in the following figure 3.15 that plainly shows the problems of the architecture for efficient floorplanning.



**Figure 3.15 HTAX Structure**

Requests from every functional unit must be routed to the arbitration logic that determines which request is granted. The result of this combinatorial arbitration logic then controls a multiplexer that selects the correct datastream that will be switched to the output of the stage. This architecture is replicated for every HTAX port, in EXTOLL's case nine times. Therefore it is important for the stages before the HTAX to intermingle with the switch logic so that their output registers can be placed ideally equidistant from all arbiters and muxes. This also means that the HTAX crossbar should not be constrained to spread from the top to the bottom of the chip. Instead its logic must be packed as close as possible. Otherwise it is highly unlikely that the long combinatorial paths will meet timing.

### - EXTOLL XBAR

The last identified critical structure is the EXTOLL crossbar. Its internal structure is depicted in the following figure 3.16.

The crossbar is architecturally separated in *inports* and *outports*. The inport contains all buffers to store packets and management information while the outports mainly consist of the multiplexer logic for the datapath and the arbitration logic. Although the EXTOLL crossbar is more pipelined than the previously discussed HTAX physical implementation is still challenging. For once, there is a large size mismatch between the inports and outports as seen in the following table 3.3 which shows that one inport due to its memory requirements occupies about 55 times the area of one outport.

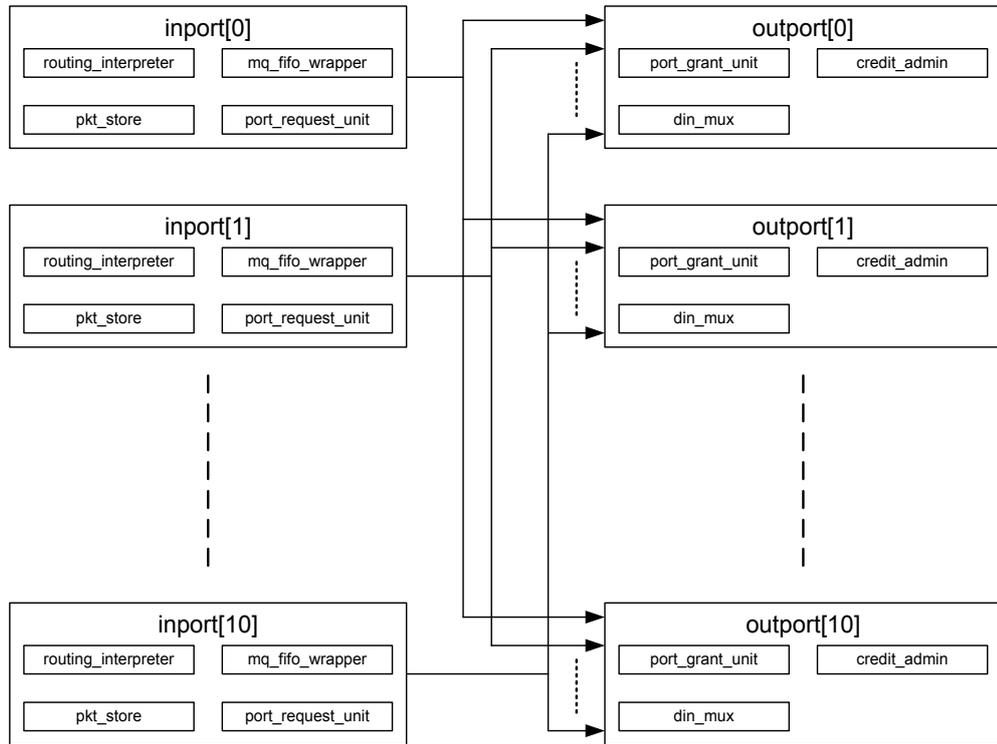


Figure 3.16 EXTOLL XBAR Structure

	Inport	Outport
Area ( $\mu\text{m}^2$ )	~ 1550000	~ 28000

Table 3.3 XBAR Area Distribution

This also means that one single outport only occupies less than 2% of the total area in an 11x11 XBAR. Each output must therefore be strategically placed in the crossbar area so that it can be reached by every inport for both the arbitration requests and also the datapaths. On the other hand it is essential to utilize the area between the serializers that is seen in figure 3.13 as much as possible which means pushing most of the crossbar logic into the gap which, on the other hand, is detrimental to the goal of placing the transitions between inport and outport as close together as possible.

In the end each case must be carefully analyzed and several floorplan iterations will have to be tried out before a final floorplan is achieved that will also allow reaching the final goal of timing closure.

### 3.5.4 Miniature Optimizations

Besides floorplanning the modules from the upper design hierarchies there are also structures at the bottom of the hierarchy that can benefit from a careful optimization process. In the following two methodologies are presented that can alleviate timing restrictions that can occur around RAMs in the design.

As mentioned in chapter 2 the RAMs are modeled so that they behave exactly as their counterparts in the FPGA design. This means that the output register stage that would normally be added at the output of a module is missing and the inserted ECC logic reduces the timing budget of the following stage. Of course, this increases the problem of wire delays because there are more logic cells in a path that must be interconnected. Thus, it seems beneficial to group the ECC logic as close to the RAM macro as possible.

This feature is supported by a relative floorplan feature that allows defining a placement relationship between different floorplan objects without knowing in advance where they will be ultimately placed.

In the case of the ECC logic this means that the gates can be tightly grouped together by a fence object and then be relatively placed next to the output pins of the RAM macro. The following TCL command places the *ECC\_logic\_grp* object relatively to the RAM instance *ram\_I* with no spacing in between:

```
relativeFPlan --relativePlace ECC_logic_grp TL ram_I BL 0 0
```

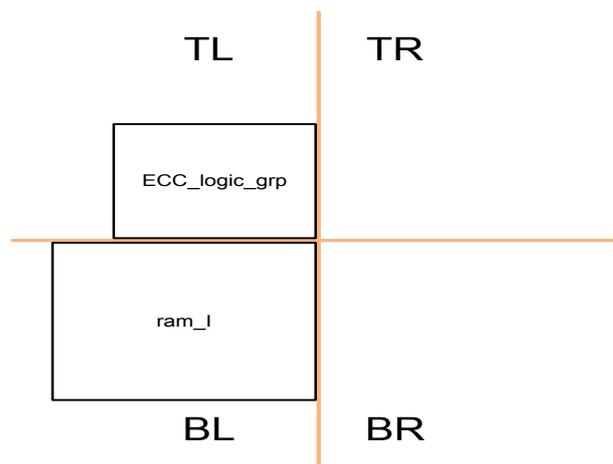


Figure 3.17 Relative Placement Example

Placement information is quadrant based with top left (TL), top right (TR), bottom left (BL) and bottom right (BR) as possible object locations. The relative placement defined in the small example code can be seen in figure 3.17.

Another example of a floorplan optimization at the bottom hierarchy level is the register based RAM. As described in paragraph 2.4 small RAMs are designed as arrays of flip-flops because they are so small that using a dedicated RAM macro would be inefficient. The following code snippet shows the module definition of such a flip-flop based RAM.

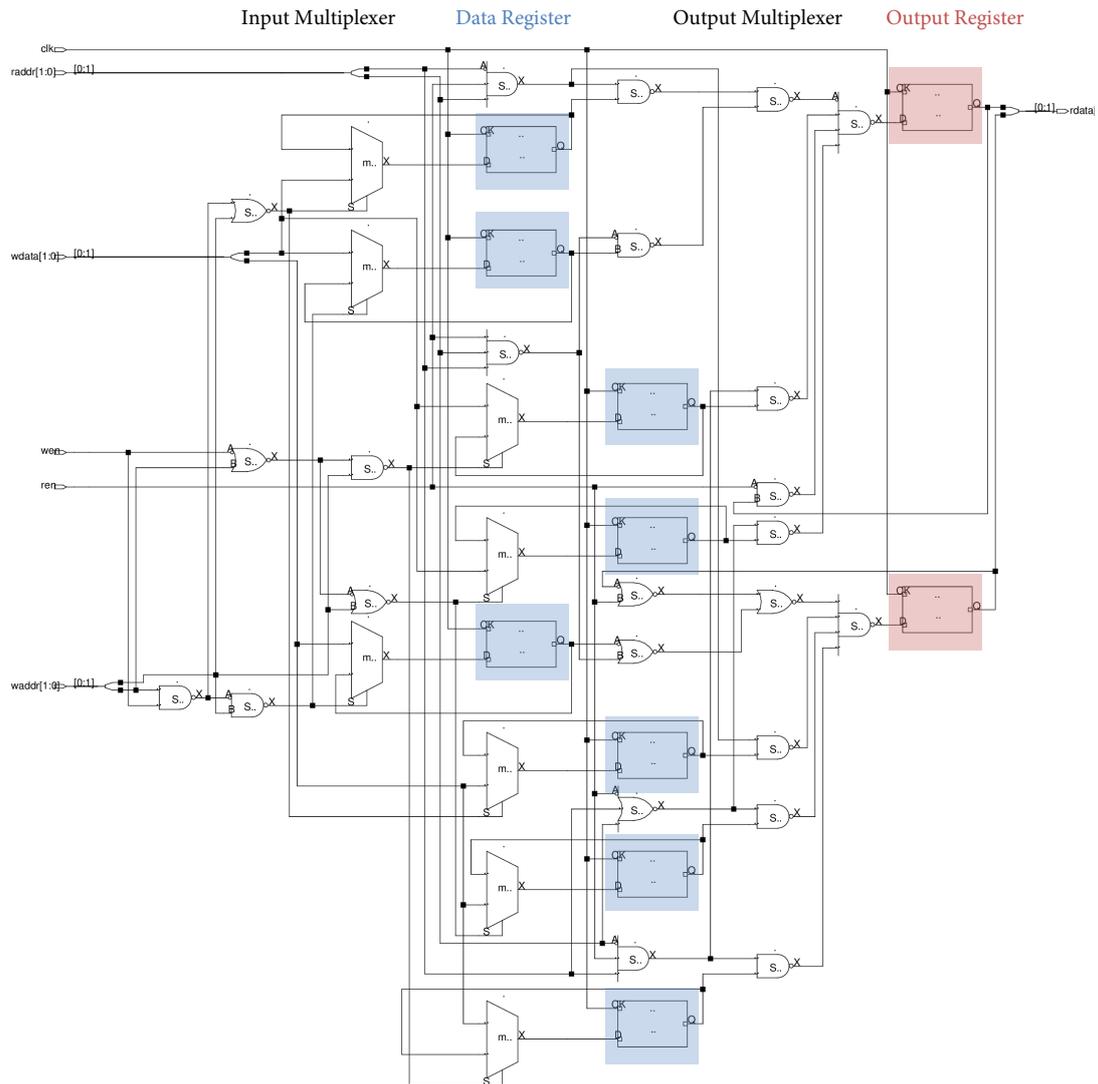
```
module reg_ram #(
    parameter DATASIZE = 2, // Memory data word width
    parameter ADDRSIZE = 2 // Number of memory address bits
) (
    input wire clk,
    input wire ren,
    input wire wen,
    input wire [DATASIZE-1:0] wdata,
    input wire [ADDRSIZE-1:0] waddr, raddr,
    output reg [DATASIZE-1:0] rdata
);
reg [DATASIZE-1:0] MEM [(2**ADDRSIZE)-1:0];

always @(posedge clk) begin
    if (wen) MEM[waddr] <= wdata;
end

always @(posedge clk) begin
    if (ren) rdata <= MEM[raddr];
end

endmodule
```

For a RAM definition with the width  $d$  and an address width of  $n$  bits a total of  $d \cdot (2^n + 1)$  flip-flops are needed. The resulting synthesized structure that can be seen in figure 3.18 is highly regular. Each flip-flop of the memory array is preceded by a multiplexer that switches between the currently stored value and the applied value at the  $wdata$  input. The switch to the  $wdata$  input only happens if both  $wen$  is asserted and  $waddr$  matches the row of the memory element. The same applies to the output register  $rdata$  which is updated with the value of the row selected by  $raddr$  when  $ren$  is asserted. These structures also scale for memories that contain more storage elements than the 4x2 array used as an example.



**Figure 3.18 Schematic of a 4x2 Register Based RAM**

Encounter supports so called Structured Data Paths (SDP) that can be used to layout such register arrays in an efficient way. It is a semi-custom design methodology that allows exact placement of single elements either at a fixed location or relative to each other.

The following TCL command, for example, aligns all flip-flops from memory row 0 in the previous example also in a single row on the chip:

```
createSdpGroup -name row0 -type row -inst reg_ram/MEM_REG[0]*
```

This means that resulting structure is a row although the elements are actually placed in columns next to each other.

The process can be repeated for all rows so that four groups exist each consisting of all memory cells for a particular row address. These groups can then be combined to an array structure:

```
createSdpGroup -name array -type column
addSdpGroupMember -group array -object {row0 row1 row2 row3}
```

In this case the defined structure is a column. Thus, each row is arranged below the previous row so that the resulting placed structure will look as follows on the chip:

<b>MEM_REG[0][0]</b>	<b>MEM_REG[0][1]</b>
<b>MEM_REG[1][0]</b>	<b>MEM_REG[1][1]</b>
<b>MEM_REG[2][0]</b>	<b>MEM_REG[2][1]</b>
<b>MEM_REG[3][0]</b>	<b>MEM_REG[3][1]</b>

Although the approach is scripting based lots of effort has to be invested to build an efficient solution. Detailed knowledge of the design or the respective module and its internal structure is a prerequisite. Therefore it only makes sense to apply the methodology in cases where the automatic placement algorithms of the tool cannot find a solution that will reach timing closure and the designer determines that the cause is an irregularly placed structure that can be optimized by hand to meet timing.



## 4 Optimization of Complex Interconnection Structures

As described in chapter 2 the EXTOLL ASIC features several external high-speed interfaces to connect to the outside world.

Interface	Signaling	Lanes	Speed	Signals	Total pins
PCIe Gen3	Differential	16	8 Gb/s	16 TX 16 RX	64
HyperTransport	Differential	18	5.2 Gb/s	18 TX 18 RX	72
EXTOLL Link	Differential	7 x 12	8 Gb/s	7 x 12 TX 7 x 12 RX	336

Table 4.1 EXTOLL High Speed Connectivity

Adding up the numbers from table 4.1 gives a total of 472 high speed pins that must be handled in the design process. This large number means that all design aspects must be carefully considered when building and optimizing the signal path from the chip to the external connector. All testing and verification efforts of the digital logic to maintain confidence are in vain if a successful and stable communication is not possible.

### 4.1 Design constraints

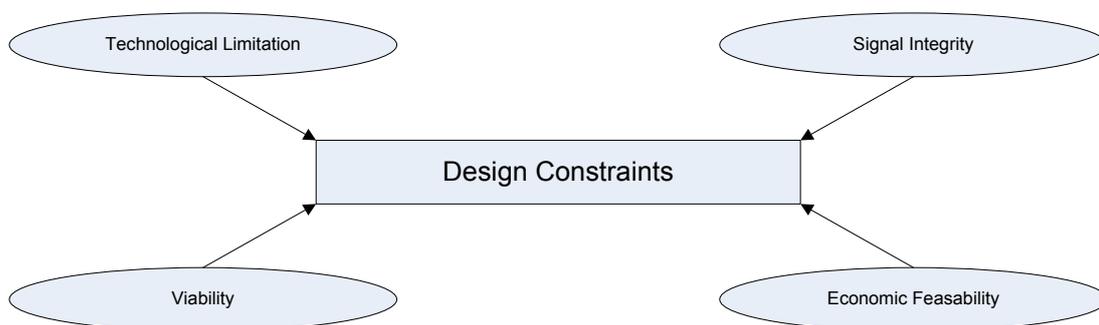


Figure 4.1 Connectivity Design Constraints

Figure 4.1 shows the design constraints associated with the optimization. In order to find an ideal solution all four constraints must be met and in turn also optimized. The following paragraphs will present these obstructions and how they are applied to the different stages in the signal path.

### 4.1.1 Technological Limitations

At several points in the design process it can happen that the designer can think of a better solution, but instead the option is not viable because it cannot be manufactured. A good example is the width of a PCB trace which is usually limited to 75 $\mu\text{m}$ , but it might be required to lower the width in order to match the impedance. The alternative is changing the material to one with another permittivity which might increase the production cost.

### 4.1.2 Signal Integrity

There are several aspects that must be kept in mind in order not to mess up signal integrity [73]. Enough distance between adjacent wires to minimize crosstalk, continuous ground reference planes, as few impedance discontinuities as possible are just some examples of good design practices that are the foundation of a clean signal transmission. Unfortunately, this might limit the available space for routing signals, for example.

### 4.1.3 Viability

It might be desirable to pack all signals together in as little space as possible. However, high density is no use if the design is not routable at any stage (e.g. in the package or on the PCB). At any case the layout must be carefully optimized early on so that as few signal crossings as possible will occur, otherwise the available routing channels might not suffice to route all connections.

### 4.1.4 Economic Feasibility

Not everything that is technologically possible should be considered. Usually, pushing the technological limits comes with a hefty price tag and it is important to find a good compromise between cost on the one side and performance on the other side. E.g. the shrinking of the trace width suggested in paragraph 4.1.1 can be done to a certain degree. Unfortunately, this is not considered to be a “standard” process any more and will result with a surcharge of up to 30% on the board price. Since this is a recurring cost for every PCB it will diminish the earnings per unit independent of volume and should be avoided.

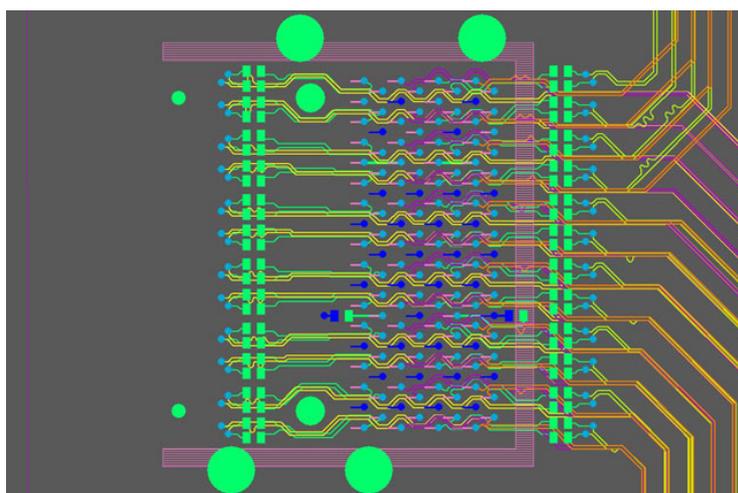


speeds of 14Gb/s, however, the available IB connectors QSFP and CXP are either limited in the number of lanes or built with huge dimensions. Since the goal is building a 3D torus network it is necessary to have six links. In order to build a card that adheres to the PCI Express Card Electromechanical Specification [75] all six connectors must fit into the slot bracket of a standard PCIe slot. This is only possible with a high-density connector which excludes almost all commercially available products from the previous table. The only connector that fits the specification is the HDI6 connector from Samtec [76]. Samtec fits two double-stacked connectors with a total height of 13.34mm in a width of 23.95mm by maintaining a pitch of  $635\mu\text{m}$  between pins. The high pins-per-volume density also led to the adaption as official connector for HyperTransport between nodes [77]. The following picture 4.3 shows the configuration with six links at the edge of a PCIe board.



**Figure 4.3 Samtec HDI6 Connector**

As seen in the following figure 4.4 the width of the connectors allows to place exactly 24 AC coupling capacitors for both the upper and lower part.



**Figure 4.4 PCB Layout of HDI6 Connector**

### 4.2.2 PCB

As seen in figure 4.4 the HDI6 connector is ordered in four rows. For a complete breakout of all signals four routing layers are needed on the PCB, one for each row. Two rows carry the TX signals of both the upper and lower part of the connector and the other two rows the RX signals. AC coupling capacitors are placed on the bottom side of the PCB, one array towards the board edge, the other behind the connector.

The PCI Express Card Electromechanical Specification defines the nominal thickness of PCIe addin cards with 1.57mm (+/- about 10% tolerance). This limits the amount of layers that can be stacked in a multi-layer board. For signal integrity reasons the high-speed signals of the external links are routed as differential striplines with continuous reference ground planes both above and below the track for proper shielding and to minimize the path for the return current.

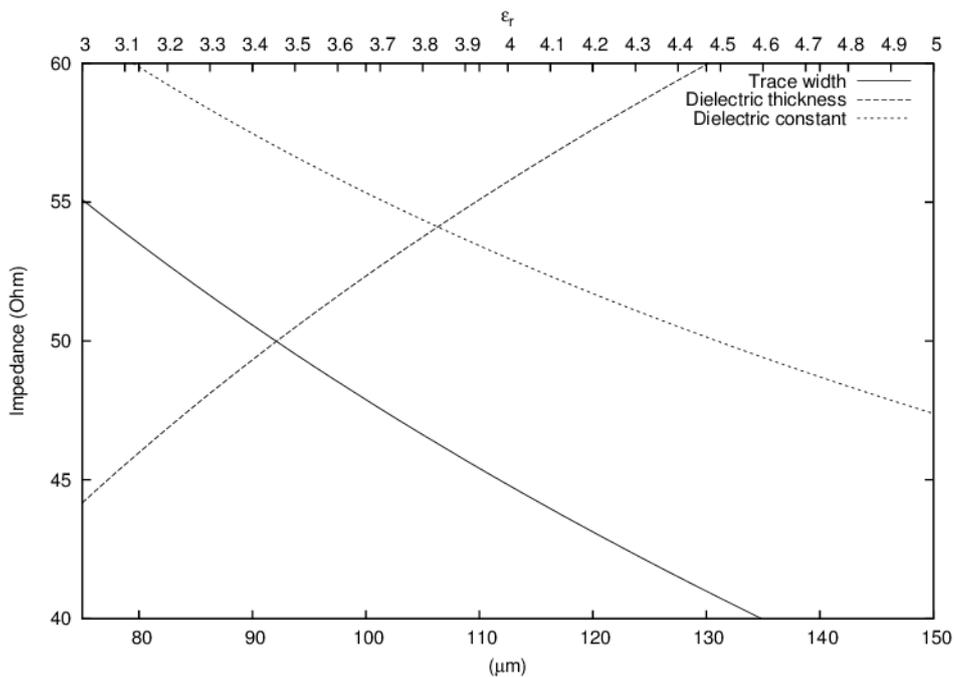
With these parameters there is not much room left to vary the structure of the PCB internally: four inner routing layers, each surrounded by ground planes and the top and bottom layer as well leave only enough space to add two more inner layers with higher thickness for power routing without violating the PCI Express guidelines. Altogether, a 14-layer board is needed to meet the routing requirements.

The final stackup also has a large influence on the impedance of the signal lines in the PCB. For accurate calculations a 2D field solver is needed to determine the impedance of a differential stripline. Nevertheless, approximations that are not far from the final result can be made with the help of some simple formulas that only need some basic parameters.

The impedance of a single stripline with a thickness  $t$  and a width  $w$  embedded in a dielectric of thickness  $b$  and a relative dielectric constant of  $\epsilon_r$  can be calculated as follows [78]:

$$Z_0 = \frac{60}{\sqrt{\epsilon_r}} \ln \left[ \frac{1.9(2b + t)}{(0.8w + t)} \right] \text{ Ohm}$$

The following figure 4.5 shows the effect that changing either the dielectric material, its thickness or the trace width has on the resulting impedance.

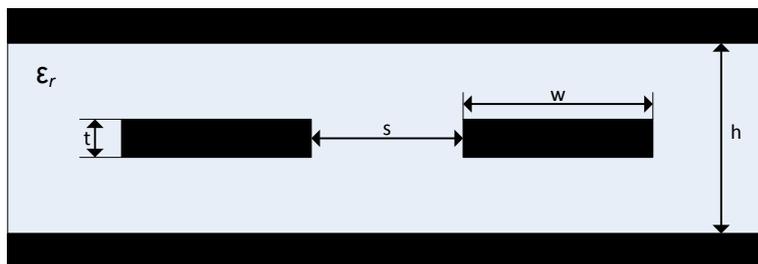


**Figure 4.5 Impact of Parameters on Impedance**

The differential impedance of an edge-coupled differential stripline with FR-4 can be approximated with the following formula [79]:

$$Z_{diff} \cong 2Z_0 \left( 1 - 0.374e^{-2.9\frac{s}{h}} \right) \text{ Ohm}$$

where  $Z_0$  denotes the impedance of a single trace without coupling,  $s$  is the distance between the differential traces and  $h$  is defined as the thickness of the dielectric between the ground planes.



**Figure 4.6 Differential Stripline**

Since the dielectric constant of “improved” FR-4 is in the range of 3.7 (may vary depending on the manufacturer) and the thickness of the dielectric as well as the thickness of traces is predefined by the stackup and the manufacturer’s guidelines the main variable that can be

## Optimization of Complex Interconnection Structures

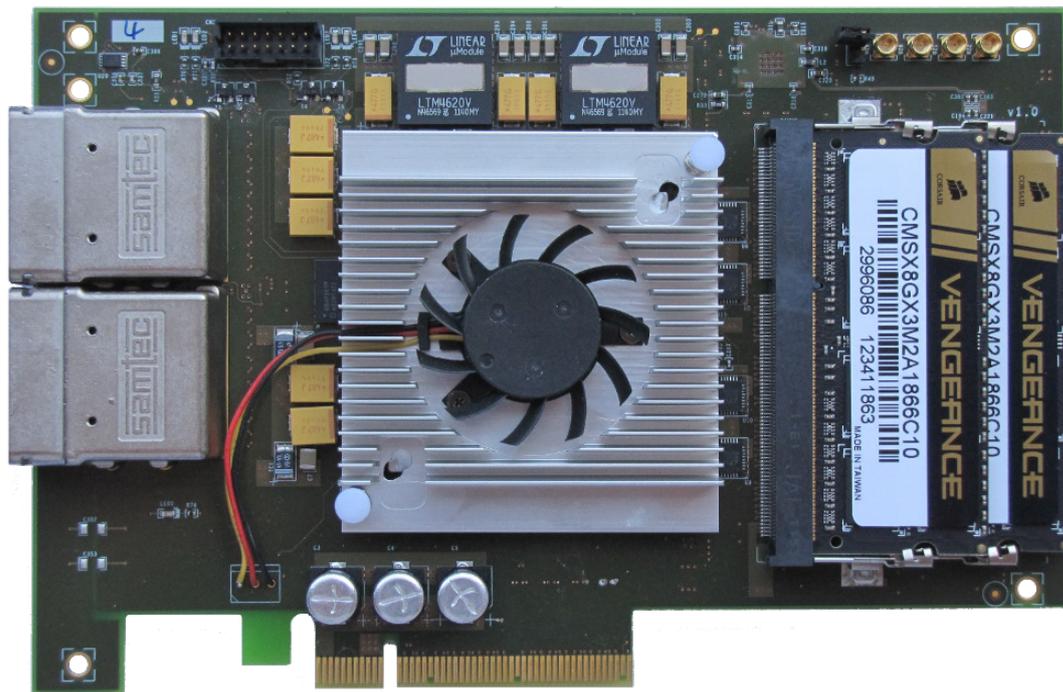
altered is the width of the trace. Many PCB companies define  $75\mu\text{m}$  as the minimum width that can be manufactured without special effort. Therefore the resulting maximum impedance that can be achieved is roughly  $55\Omega$ . The differential impedance is twice the single trace impedance in the case that no coupling occurs. For coupled traces, however, the differential impedance decreases the nearer the traces come together. As a final result, differential routing in the EXTOLL board stackup was defined by two traces with  $75\mu\text{m}$  width that are edge-coupled with a distance of  $125\mu\text{m}$ , resulting in a differential impedance of about  $100\Omega$ . The outcome is close enough to the desired values (since the parameters are not always 100% accurate) that PCB manufacturers that offer controlled impedance for their boards are able to tweak the parameters marginally to produce the board without problems. The following figure 4.7 shows an example stackup with the previously defined parameters.



Figure 4.7 Example PCB Stackup

However, the exact configuration may slightly vary depending on the manufacturer and their available core and prepreg offerings.

In order to test the routability and to uncover eventual signal integrity issues a prototype board was designed that featured two 12x capable double-stacked HDI6 connectors driven by a Xilinx Virtex7 FPGA that is able to generate traffic at speed of up to 8.5Gbit.



**Figure 4.8 Aspin Test Board**

With the Aspin prototype board and the two Virtex6 based boards presented in paragraph 2.7 connectivity between chip and connector has been explored sufficiently. The boards also gave valuable insight for developing the best power strategy for the final EXTOLL board.

### **4.2.3 Package**

High pin count devices must be packaged as Ball Grid Arrays (BGAs). Because of the arrangement of the solder balls as a grid array on the bottom side of the package a very high pin density per area can be achieved.

BGAs also have other advantageous properties [80]:

- During the soldering process the package itself can compensate a misalignment of up to 50% due to the surface tension of the solder that pulls the chip to the correct position.
- The larger number of balls between package and PCB and their composition facilitate the heat transfer due to a lowered thermal resistance. The thermal resistance directly correlates to the maximum allowed power dissipation:

$$P_{max} = \frac{\Delta T}{\Theta_{JA}}$$

$\Delta T$  is defined as the difference between the junction temperature (about 125°C) and the ambient temperature. The junction-to-ambient thermal resistance  $\Theta_{JA}$  is the sum of all thermal resistances. A lower thermal resistance of the package means that the chip can either dissipate more power or that other cooling solutions can be designed more conservatively.

- Because the critical signal lengths in a BGA package can be greatly reduced the lead inductance shrinks which has a positive effect on signal integrity which makes a BGA an ideal candidate for high speed chips like EXTOLL.

The JEDEC publication 95 [81] defines a high number of different BGA packages. One of their differentiating parameter is the pitch which defines the distance between the center points of two adjacent balls on the bottom side of the package.

FPGA vendors which can be seen as producers of commodity devices offer their chips in packages with a 1.00mm pitch with only few exceptions that have a pitch of 0.8mm.

A smaller pitch obviously has the advantage of a smaller package if the number of pins stays the same or more pins than before can be accommodated on the same area. Especially chips for mobile devices are nowadays produced with a pitch as small as 0.4mm. However, the on-going miniaturization leads to technical problems:

- A smaller pitch also leads to smaller solder balls which are usually weaker in resisting thermal cycling fatigue and might not be as reliable as the larger balls of devices with a larger pitch [82].
- The higher connection density also poses a challenge for the PCB design. Signals from all over the package must be routed to other devices or connectors. Before this can happen, a so-called breakout must be done to route all signals from the bottom of the chip to the outside area. The common approach consists of placing a via diagonally across a pin and perform the breakout on another layer because the top layer does not have enough routing channels for all signals (except for very small

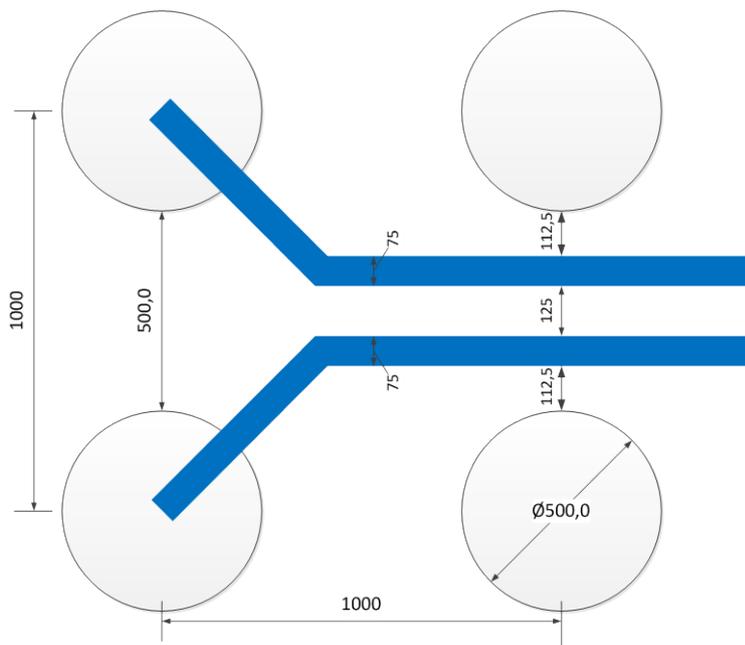
## Optimization of Complex Interconnection Structures

---

chips). However the space between these vias that are drilled through the complete PCB is also limited so that only a certain amount of signals can be routed below the BGA on a single layer. If the pitch is very small only a small wire will fit in a routing channel. Most likely this will lead to an impedance discontinuity because spacing rules limit the wire to a certain width that will not lead to an optimum impedance. Differential routing will definitely not be possible below a fine pitch BGA.

Of course, the rule that finer structures are always more expensive also applies to the miniaturization of BGA packages.

In the case of EXTOLL a 42x42 BGA package was defined to be sufficient to handle the signaling and power requirements of EXTOLL. With 1764 balls and a proposed pitch of 1.00mm the package exhibits the same properties as the Virtex6 LX240 that is assembled on the Ventoux and Galibier prototype boards. The FPGA prototypes already showed that signals can be routed as differential pairs through a routing channel below the chip as shown in the following figure 4.9.



**Figure 4.9 Differential Pair BGA Breakout**

The most critical part of the pin arrangement on the package is the location of the serializer pins. As defined in the previous paragraphs four inner routing layers are available due to the stackup which also matches the number of layers to route to a double-stacked connector. The

best way to route a high-speed signal is over a continuous ground reference plane without layer changes so that the return path is uninterrupted. Therefore, the obvious solution is to use one layer for RX of one link, route TX on another signal layer and do the same for the second link on the same connector. The serializers are already placed in rows on the die which translates to a similar arrangement on the package. Due to the high number of serializers a link consists of two rows (one for the P pins of the differential pair, the other for the N pins) and 24 columns (12 RX and 12 TX). The previous figure showed that only one differential pair can be routed between vias under the FPGA. This means that twelve routing channels must be available per layer to route either RX or TX of one link. The following figure 4.10 details the breakout pattern for this problem. The two foremost differential pairs are directly routed out of the BGA area (albeit on two different layers). The next two pairs share the adjacent routing channel. In the middle of the row the differential pairs will overlap with the signals of the second connector. These signals, however, use the other two remaining layers to route independently of the other signals. The rearmost differential pairs will make use of the routing channel which lies next to the pins belonging to the next connector. Since the remaining pins will be either unconnected or connect to ground or power planes with their breakout vias it does not matter that the routing space below the FPGA is completely utilized in the serializer area.

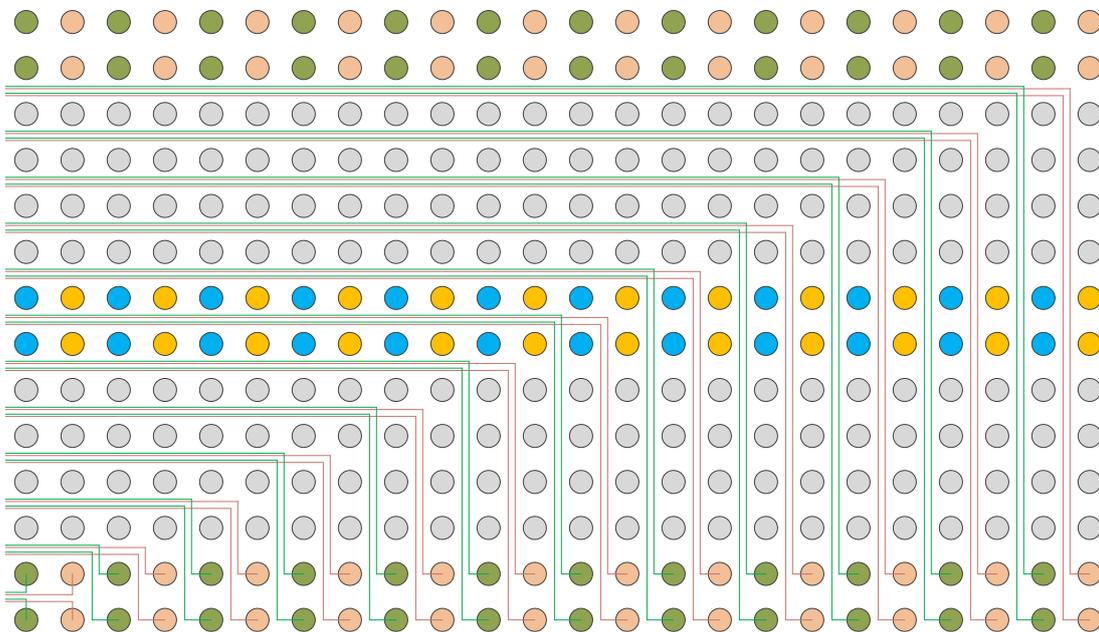


Figure 4.10 A 12x Serdes Breakout

EXTOLL's package will be manufactured by Kyocera SLC Technologies [83] that offer an organic substrate with high density build up that is especially suited for high pin count BGA devices with many high-speed signals. The package is designed in a 4-4-4 stackup [84] configuration with a 4 layer core surrounded by 4 build-up layers on each side. Final substrate height will be about 1160 $\mu\text{m}$ .

### 4.2.4 Die

Dies can be attached to a package either as flip chip [85] or with wirebonds. Because of the large number of pins required and lots of high speed signals that greatly benefit from the lowered inductance of a flip chip attachment EXTOLL is designed as flip chip which logically fits to the choice of a BGA package since the same arguments apply that imply that a BGA is a superior choice for high-speed signaling.

For flip chip the pins on the die are placed in an array on the top side of the chip. The same pattern is also present on the package substrate. In order to bond die and package together the die is flipped so that the pins face the substrate. This process was introduced by IBM in 1964 [86] and called *Controlled Collapse Chip Connection* (C4) which is still a common acronym for flip chip technology.

EXTOLL's die size is mainly determined by the size of the serializer macros. Likewise the location of macros and I/O cells already predefines the general location of the associated bumps so that the floorplan directly matches to the external connectivity. This avoids long routing delays on the redistribution layer (RDL) that can potentially deteriorate signal integrity.

The allocated die area implies minimum rules for bump size and pitch according to the TSMC reference manual [87] for the technology. Bump pitch and die size in turn determine the size of the bump array. The number of available bumps on the die is larger than the available balls on the package since it is common to define a complete array mainly due to mechanical reasons to prevent the die from breaking when it is attached to the package. This means that many bumps do not have a signal assigned. These bumps can either be left as dummy bumps with no connection to the chip or used as additional ground / power contacts to improve the power distribution network.

### 4.3 Automatic Generation

Changes in the signal arrangement can happen frequently, especially in the starting phase of a design. While earlier changes tend to be more extensive, later updates will usually only shift the position of very few pins.

These changes always affect a set of different tools that use different databases to store this information. Changes on the die must be kept in sync between the backend tool and the package design tool while changes on the BGA side must be reflected in the PCB tool.

Therefore, the best approach to avoid any inconsistencies was the development of a small TCL script that generates assignments for all tools from a global database while taking care of the different tools' peculiarities:

- Depending on the tool the pin location must either be done by absolute x/y positions or by row/column definition.
- While the backend tool numbers rows and columns with digits the package designer relies on the JEDEC definition.
- The backend tool needs a clear distinction between signal and power/ground pins.

The script takes care of all these corner cases and generates a definitions text file for the package designer that can be directly imported and a TCL script for the backend tool that assigns all pins after it is sourced.

Since many structures appear with a regular pattern (like the serializer arrangement or the location of power/ground pins) and with names that can also easily be composed automatically the maintenance effort for the script is rather low while reducing risk of a mismatch in the databases at the same time.

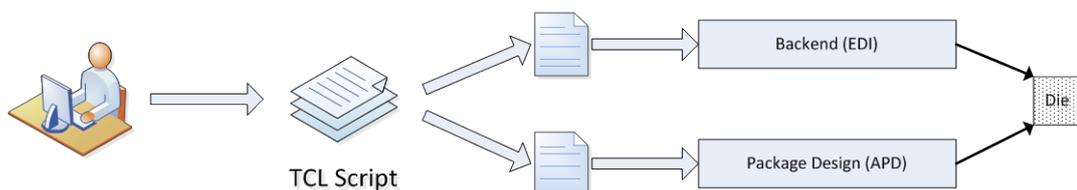


Figure 4.11 Scripting Approach

### 4.4 SI Analysis

For every high-speed design an accurate signal integrity analysis is of utmost importance. While good design practices help to achieve building a system that will work in most cases, especially for multi-GHz signals the simulation of the traces with a field solver cannot be avoided.

While the package design was performed with Cadence Design System's *Allegro Package Designer* [88] all simulations were accomplished using Agilent's *Advanced Design System (ADS)* [89].

This paragraph will give a short overview of the simulations for the high-speed differential traces for the EXTOLL links that are designed to run at speeds of up to 10 Gbit/s.

The following figure 4.12 shows the simulation setup for a single channel. The testbench consists of a pseudo random generator that drives a differential CML driver. The receiver side is terminated to the current mode voltage  $V_{CM}$  inside the serializer. The data from the differential receiver is then analyzed in the connected probe. In between the channel model of the extracted package traces is inserted in a first step. For further analysis the rest of the signal path like PCB traces and connector models can also be inserted to simulate the whole system to get an authoritative answer regarding the signal integrity.



**Figure 4.12 Single Channel Simulation Setup**

The following figure 4.13 shows an eye diagram that was measured after the signal has crossed the package. The BGA balls, however, have not been included in the simulation. The simulation has been run at 10 Gbps with an output driver voltage swing of 800mV across a package trace with a length of 25.6 mm. No equalization was performed.

The eye diagram with a width of 98.2ps and a height of 595mV shows pretty good results considering the high signal rate. However, the trace is rather short and there are no impedance discontinuities. The eye diagram at the end of the complete channel model will look much worse. For a detailed analysis of signal integrity problems and further simulations kindly refer to [90].

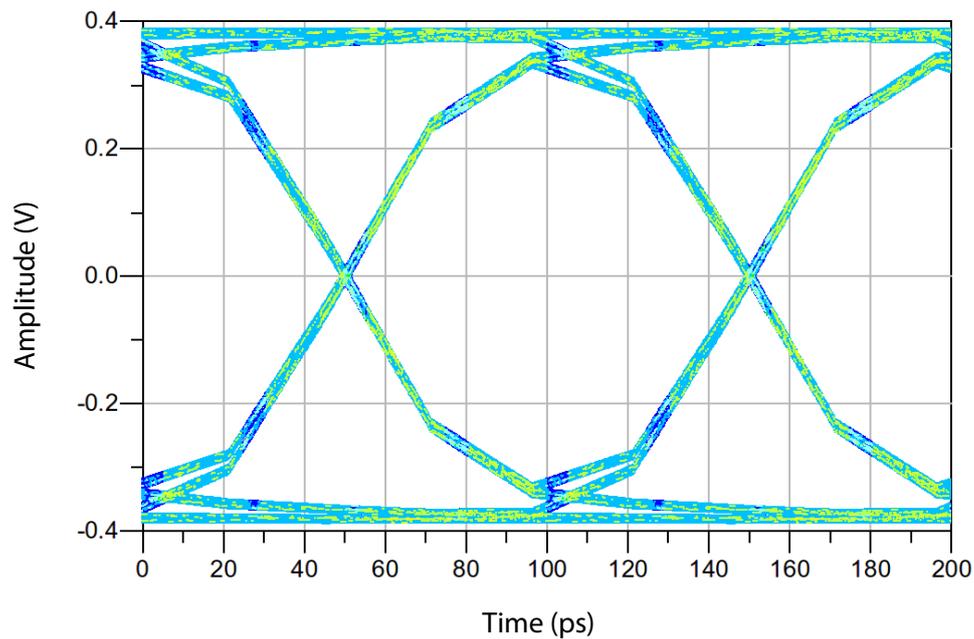


Figure 4.13 Eye Diagram at 10 Gb/s

### 4.5 PDN Design

Designing a sufficient power distribution network is as important as designing for optimal signal integrity. A core supply that looks like the one in the following figure 4.14 will cause unexpected problems during the device's operation and the chip might fail to work correctly.

The goal of an efficient PDN is to supply the same voltage all over the chip with an allowed noise tolerance (ripple) and to reduce the effect of ground bounce. Both goals are related in some way and are affected by unwanted inductance in the current path. Ground bounce [91] which is also called simultaneous switching noise (SSN) occurs during the switch from 1 to 0. The current flowing from the load capacitor in combination with the inductance of the current path generates a voltage which can be seen as ground bounce on a scope.

$$V = L \frac{di}{dt}$$

If only few circuits switch at the same time the induced voltage is minimal, however, when many circuits switch simultaneously the effect is measurable.



**Figure 4.14 Example of an Insufficient Power Supply**

On the other hand, the current going through the power network to the chip causes a voltage drop (IR drop) that is caused by the resistance of the network. However, the power distribution network does also have inductive and capacitive qualities leading to an impedance that is dependent on the frequency so that the resulting voltage is a function of the complex impedance of the PDN and the current flowing through it.

Thus, the goal of the PDN design is to achieve a target impedance that is low enough that the voltage drop caused by it is below the specified ripple specification  $r$ .

$$Z < \frac{rV_{DD}}{I_t}$$

In cases where the transient current  $I_t$  is not available an approximation of about 50% of the peak current is a good estimation.

For FPGA designs a concrete PDN analysis is not really necessary since the FPGA vendors have already run extensive simulations and provide elaborate guidelines on the number and values of the required decoupling capacitors. Especially for first chip releases these recommendations are exceedingly conservative so that adhering to the guidelines should suffice for a smooth operation.

For EXTOLL such data is not available for now so that an accurate simulation of the power distribution network using a SPICE simulator is recommended. Overdesigning the network to achieve a lower impedance is always a viable option to build in some safety, however, additional components required for this step will cause higher costs.

### 4.6 Results

The next paragraphs present an overview of EXTOLL's final package considering the constraints for the implementation that were analyzed in previous sections.

#### 4.6.1 EXTOLL I/O

The following tables 4.3 to 4.10 show a list of all I/O signals of EXTOLL. The pins are divided in logical groups combining signals together that belong either to a certain standard or a specific application.

Pin Name	Description
HT_RX_CLK{H,L}_{P,N}	HT RX CLK differential pairs
HT_RX_{P,N}_{0..15}	HT RX CAD differential pairs
HT_RX_CTL{H,L}_{P,N}	HT RX CTL differential pairs
HT_TX_CLK{H,L}_{P,N}	HT TX CLK differential pairs
HT_TX_{P,N}_{0..15}	HT TX CAD differential pairs
HT_TX_CTL{H,L}_{P,N}	HT TX CTL differential pairs
HT_REFCLK_{P,N}	HT reference clock differential pair
HT_PWROK	HT PWROK
HT_RESET_N	HT Reset
HT_LDTSTOP	HT LDTSTOP
HT_CALTX	HT TX calibration resistor Connect an external 1.2K precision resistor between HT_CALTX and HT_CALTXRTN.
HT_CALTXRTN	HT TX calibration resistor Connect an external 1.2K precision resistor between HT_CALTX and HT_CALTXRTN.
HT_CALRX	HT RX calibration resistor Connect an external 1.2K precision resistor between HT_CALRX and HT_CALRXRTN.
HT_CALRXRTN	HT RX calibration resistor Connect an external 1.2K precision resistor between

HT_CALRX and HT_CALRXRTN.
---------------------------

**Table 4.3 HyperTransport Interface Pins**

Besides the 16 CAD, 2 CTL and 2 CLK signals for both RX and TX that transfer link data, control information and a clock for each 8 bit sublink this group also includes three sideband signals that are defined in the HyperTransport protocol.

Both the HT\_PWROK and HT\_RESET\_N signal control the reset sequence of the HT link. The HT\_PWROK signal can be viewed as a cold reset for the HT system and indicates that both the power and the clocks are running and stable. HT\_RESET\_N on the other side can be seen as a warm reset that resets the HT link. After powering up the initialization sequence consists of a deassertion of the HT\_PWROK signal by the host system followed by a deassertion of HT\_RESET\_N a short time later. The HT\_LDTSTOP signal is a power management signal for the HyperTransport link that can both stop (upon assertion) and resume (upon deassertion) link operation. Usage of both HT\_RESET\_N and HT\_LDTSTOP is commonly limited to changes in the link configuration after the BIOS has determined both the host's and the add-on card's capabilities.

Additionally, the 200 MHz reference clock that is the base clock for all links in the HyperTransport system is fed into the chip through the HT\_REFCLK differential pair.

For transmitter and receiver impedance calibration two pin pairs (HT\_CALTX/HT\_CALTXRTN and HT\_CALRX/HT\_CALRXRTN) are provided that must be connected to an external precision resistor.

Pin Name	Description
PCIE_RX_{P,N}_{0..15}	PCIe RX differential pairs
PCIE_TX_{P,N}_{0..15}	PCIe TX differential pairs
PCIE_REFCLK_{P,N}	PCIe reference clock differential pair
PERST_N	PCIe Reset

**Table 4.4 PCI Express Interface Pins**

The PCI Express pin group consists of 16 lanes for data transfer between root port and endpoint. Additionally, the PCIe reference clock with a nominal frequency of 100 MHz (with an allowable deviation of  $\pm 300$ ppm) is provided through the PCIE\_REFCLK differential pair. Like the HyperTransport reference clock the PCI Express reference clock allows for spread spectrum clocking (SSC).

PERST\_N is the only reset signal provided in a PCI Express environment. Its meaning is analogous to HT\_PWROK, i.e. it is deasserted as soon as both the power and the clocking is completely stable.

Pin Name	Description
L[0..6]_RX_{P,N}_[0..11]	RX links
L[0..6]_TX_{P,N}_[0..11]	TX links
L[0..6]_REFCLK_{P,N}	External link reference clock input
L[0..6]_CBL_DET_N	Cable detect for links (internal pullup)

**Table 4.5 Network Interface Pins**

Although table 4.5 is only sparsely populated it covers a large number of pins. Each of the seven network link consists of 12 different pairs for each receive and transmit as well as a reference clock input. The reference clock input, however, does not have to be connected necessarily since the normal mode of operation is supplying the reference clock on-chip as described in section 2.6.

A simple state signal with an internal pullup resistor for cable detection that is pulled down to ground as soon as a cable is inserted is also provided once per link.

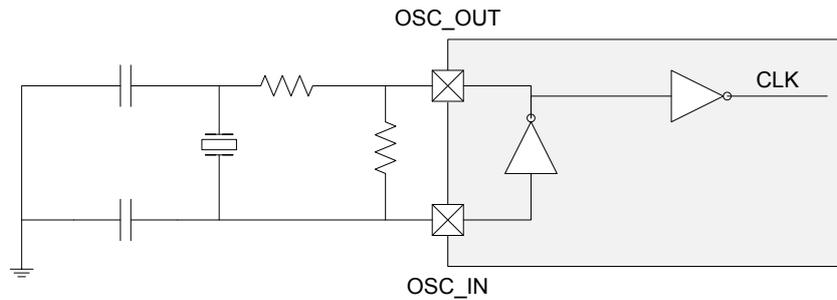
Pin Name	Description
OSC_IN	Oscillator input, crystal connection, or 25MHz ext. input
OSC_OUT	Oscillator out, crystal connection
EXT_CLK_IN_{P,N}	External CML clock input, high speed input
DBG_CLK_OUT_{P,N}	Debug clock leaf output
DBG_PLL_CLK_{P,N}	Debug PLL output clock
DBG_PLL_REFCLK	PLL reference clock debug output
PLL_LDO_OUT	PLL internal LDO output pin, connected to at least 4.7μF. It can also be directly connected to 1.2V for PLL core. All blocks of the PLL are directly connected to this voltage.
PLL_BG_OUT	Bandgap output current, which is linearly proportional to the temperature. The pin is pulled to GND with a weak pulldown resistor.

**Table 4.6 Clocking Pins**

In the clocking section all pins are grouped that are related to either the internal PLL or defined as backup alternatives. The OSC\_IN / OSC\_OUT pins are used to build an oscillator

circuit (as seen in the following figure 4.15) that generates the 25 MHz reference clock for the PLL which is the foundation of the core EXTOLL clock.

In order to assure correct functionality a high-speed differential clock input EXT\_CLK\_IN is also provided for injecting the core EXTOLL clock directly, thus circumventing the PLL.



**Figure 4.15 Oscillator Circuit**

For initial bringup three clock debug ports are connected that will output the clock during different stages in its generation. During normal operation these pins may be unconnected.

Pin Name	Description
EXTOLL_PON_RES_N	EXTOLL global power-on reset (internal pullup)
PCIE_IF_SELECT	Host interface select signal. (0: HT, 1: PCIe (internal pullup), static)
INT_CLK_SELECT	Clock source select signal. (0: external clk, 1: internal clock (internal pullup), static)
PCIE_EP_ENABLE	Switch between PCIe rootport and endpoint capability. (0: RP, 1:EP (internal pullup), static)
PATCH_FROM_FLASH_N	Configures the patch flash mechanism (0: patch, 1: no patch (internal pullup), static)
DBG_MODE_EN_N	Configures the GPIO interface as debug output. (0: debug mode on GPIO, 1: std. operation (internal pullup))

**Table 4.7 Configuration and Reset Pins**

The pins in table 4.7 are used to hard wire configuration options and control the reset of EXTOLL.

The PCIE\_IF\_SELECT signal controls which of the two host interfaces is active. The default setting is PCI Express since that will be the most common setting. Furthermore, the PCI

Express mode can be operated either as endpoint (which is the default) or as rootport. This feature can be selected with the PCIE\_EP\_ENABLE pin.

The PATCH\_FROM\_FLASH\_N signal is needed to enable the patch process from an SPI flash through the debug port. The default setting is that no patching is performed, only if it turns out that there are incorrect values in the register file it is required to set the signal.

In the case of a non-functional PLL or another malfunction in the clocking circuit a backup clock coming from EXT\_CLK\_IN can be used by setting INT\_CLK\_SELECT.

EXTOLL\_PON\_RES\_N is EXTOLL's global reset signal that will be deasserted as soon as external power and clocking is stable and normal operation can commence.

Pin Name	Description
I2C_SLAVE_SCL	I <sup>2</sup> C slave debug port clock signal
I2C_SLAVE_SDA	I <sup>2</sup> C slave debug port data signal
I2C_SLAVE_ADDR_SEL	I <sup>2</sup> C slave address selection switch. Defines the LSB of the I <sup>2</sup> C slave address.
I2C_MASTER_SCL	I <sup>2</sup> C master clock signal
I2C_MASTER_SDA	I <sup>2</sup> C master data signal
SPI_Q	SPI master data input
SPI_CLK	SPI master clock
SPI_D	SPI master data out
SPI_CS_{0,1}	SPI chip select (patch flash or optional device)
GPIO_{0..15}	General Purpose I/O (shared with SI[0..7]/SO[0..7] during scantest)

**Table 4.8 Miscellaneous Pins**

Most of the signals in table 4.8 are used to connect to the debug port (see paragraph 2.5).

An I<sup>2</sup>C master that is internally controlled through the register file is reachable through the I2C\_MASTER\_SCL/SDA pins that control the clock and data signal of the interface. The same set of signals is available for the I<sup>2</sup>C slave in the debug port that can be used to read debug information from the register file during operation. In order to operate two separate chips in a common I<sup>2</sup>C chain (as intended for the DEEP system) the I2C\_SLAVE\_ADDR\_SEL pin can be used to set the least significant bit of the slave's address so that both devices can be addressed separately.

## Optimization of Complex Interconnection Structures

---

In order to read and write from an SPI flash as needed for the debug port a set of common pins (SPI\_Q, SPI\_CLK, SPI\_D) is available. Two separate flash devices can be operated that are selected through the two chip select signals (SPI\_CS).

A set of GPIO pins is also available. These can be accessed through the register file and can be used to either read the status information from an external device or control another component on the board, e.g. an LED. The GPIO interface can also be used to output debug information if the debug mode is enabled as shown in the previous paragraph. If the debug mode is enabled the lower eight pins of the bus output a set of debug signals that can help verifying the operation of the chip. These signals consist of:

- EXTOLL global reset
- Status signal that patching from the patch flash has been completed
- A trigger that a packet was either sent to or received from the host on a specific virtual channel (posted, nonposted, response)

The third mode of operation of the GPIO pins is available during scan test. In this case half of the pins operate as scan input, the other half as scan output to access the internal scan chains of the chip.

Pin Name	Description
TDI	Common JTAG data input
TCK	Common JTAG test clock
TMS_0	Memory BIST JTAG test mode input
TDO_0	Memory BIST JTAG data output
TMS_1	Serdes JTAG test mode input
TDO_1	Serdes JTAG data output
TRST_N	Testlogic reset (JTAG TAP, TDR and memory BIST)

Table 4.9 JTAG Interface Pins

All JTAG [92] pins are test related and do not have to be connected for normal operation. There are two separate JTAG controllers inside EXTOLL as shown in the following chapter 5 in figure 5.8. The two controllers share a common set of signals, namely TDI for data input, TCK as clock and a common reset TRST\_N. JTAG data output TDO and test mode select TMS that controls the state machine of the controller is implemented separately.

Pin Name	Description
DFT_TM_N	Dedicated test mode signal. NC for normal operation.
DFT_SE_N	Dedicated shift enable during scantest. NC for normal

	operation.
<b>DFT_CLK</b>	Dedicated test clock. NC for normal operation.
<b>DFT_OPCG_EN</b>	Enable OPCG pulses during scantest. NC for normal operation.
<b>DFT_OPCG_LD_CLK</b>	To load OPCG configuration register. NC for normal operation.
<b>DFT_OPCG_TRIGGER</b>	Trigger pulses. NC for normal operation.
<b>DFT_COMP_EN</b>	Enable scan chain compression. NC for normal operation.
<b>DFT_SPREAD_EN</b>	Enable scan chain decompression. NC for normal operation.
<b>DFT_MASK_LOAD</b>	Enable scan chain masking logic mask loading. NC for normal operation.
<b>DFT_MASK_EN</b>	Enable masking logic. NC for normal operation.

**Table 4.10 Dedicated DFT Pins**

All signals in table 4.10 are used for testing the chip and do not have to be connected for normal operation. In order to switch to test mode the DFT\_TM\_N signal must be asserted. Data is then shifted into and out of the internal scan chains through the GPIO pins with the DFT\_SE\_N signal and clocked with the dedicated DFT\_CLK which will only run at a moderate speed of about 50 MHz. Scan compression and decompression can be enabled with the DFT\_COMP\_EN and DFT\_SPREAD\_EN signals. In order to improve the efficiency of the compression logic unknown (X) values can be masked out. Masking will be enabled through the DFT\_MASK\_EN signal. In order to share the test clock with the masking clock (instead of providing a separate input) the DFT\_MASK\_LOAD input is used. Upon assertion the signal gates the clock to the design except the mask register so that only the mask register will be loaded.

On-Product Clock Generation (OPCG) to enable at speed testing is controlled through three pins. OPCG will be configured by loading configuration registers clocked with DFT\_OPCG\_LD\_CLK. The OPCG logic will then be enabled by asserting the DFT\_OPCG\_EN signal. In order to advance the design by one high-speed clock cycle the trigger signal DFT\_OPCG\_TRIGGER is used to generate pulses.

### 4.6.2 EXTOLL Supply

A large part of EXTOLL's package is taken up by power supply pins which will be presented in the following.

Pin Name	Description
VDD_CORE	EXTOLL logic core voltage @ 1.0V
GND_CORE	Common digital logic ground.
VDD_IO	CMOS I/O voltage @ 3.3V or 2.5V
GND_IO	CMOS I/O ground.

**Table 4.11 Digital Core and I/O Supply**

TSMC's 65nm GP process requires a core supply of 1.0V which is supplied via the VDD\_CORE pins distributed across the chip. The I/O voltage can be either 3.3V or 2.5V and is limited to the I/O islands in the chip. Each of these I/O islands could be run with a different I/O voltage; however, since all external components are able to run at 3.3V there is only a single I/O voltage which in turn also simplifies the design of the package. The ground pins of the core and I/O can be tied together since the I/O cells only operate at a low speeds and are not susceptible to increased noise.

Pin Name	Description
HT_VDDHT	HT PHY digital voltage @ 1.1V
HT_VSSHT	HT PHY digital ground.
HT_VDDC	HT PHY digital core level voltage @ 1.0V (probably 1.1)
HT_VSSC	HT PHY digital core level ground.
HT_VDD18	HT bias supply for RX termination and PLL supply @ 1.8V
HT_VDDRX	HT RX analog clean voltage @ 1.1V
HT_VDDTX	HT TX supply voltage @ 1.2V
HT_VSSA	HT PHY common analog ground.

**Table 4.12 HyperTransport Supply**

A whole set of supplies is needed to operate the HyperTransport PHY. Both receiver and transmitter need a clean analog supply, running at 1.1V (HT\_VDDRX) and 1.2V (HT\_VDDTX) respectively. An analog ground that is separated from digital ground must also be provided (HT\_VSSA). The digital core logic is supplied through HT\_VDDHT and HT\_VDDC, each coupled with ground pins (HT\_VSSHT and HT\_VSSC). For performance reasons these supplies are overdriven to at least 1.1V. Eventually, they must be increased up to 1.25V for correct operation at the highest possible speed. At last, 1.8V are supplied through HT\_VDD18 for the PLL and RX bias supply.

Pin Name	Description
VDD_PLL	PLL supply voltage @ 1.8V

<b>GND_PLL</b>	PLL analog ground.
<b>PLL_IREF_IN</b>	PLL reference current (100uA) (optional fallback)
<b>VDD_OSC</b>	Crystal oscillator I/O voltage @ 3.3V
<b>GND_OSC</b>	Crystal oscillator I/O cell ground.

**Table 4.13 Clocking Supply**

The PLL is supplied through an individual VDD\_PLL pin. Both the bandgap and the internal LDO are connected to this voltage. An analog ground pin is also required. In the case of a non-working bandgap the reference current required for the PLL can also be supplied through a separate PLL\_IREF\_IN pin.

At last, the I/O bank containing the oscillator cell is supplied separately (VDD\_OSC) from the rest of the I/O cells to avoid external noise that will influence the jitter characteristics of the clock in a negative way. A coupled ground pin (GND\_OSC) concludes the clocking supply.

<b>Pin Name</b>	<b>Description</b>
<b>TC_VCCA1</b>	Serializer analog voltage @ 1.0V
<b>TC_VCCA2</b>	Serializer analog voltage @ 1.8V
<b>TC_VCCD</b>	Serializer digital voltage @ 1.0V
<b>TC_VSSA</b>	Serializer common analog ground.
<b>TC_VSSD</b>	Serializer digital ground

**Table 4.14 Serializer Supply**

Besides the digital core most power will be consumed in the serializers in EXTOLL. The serializers need two different analog voltages running at 1.8V and 1.0V respectively and a digital voltage at 1.0V. Of course, both the analog and digital supply also require a separate ground.

### **4.6.3 Constraints and Efficiency**

Appendix C shows the layout and mechanical diagrams of EXTOLL's package.

In summary, one can say that it is important to have a global view on the whole system and optimize all corners in parallel. Unfortunately, up to now there is no tool available that can support the design process. Existing solutions only cover partial aspects like the integration of backend and package development, but they still require lots of manual work. This means that a lot work has to be done traditionally with pencil and paper to develop a coarse

## Optimization of Complex Interconnection Structures

---

approach and refine the different steps later. Although this cannot be done automatically it can be supported with a scripting approach to avoid costly mistakes.

In the overall EXTOLL system everything fits together nicely, for each part of the signal path a satisfactory result could be found without having to resort to an inferior solution due to external constraints. The final package adheres to the constraints that were defined in the previous sections and constitutes the best solution to connect all interfaces.

## 5 EXTOLL Test

ASIC production is not perfect. Although the production process takes place in a clean room there is still a possibility that small dust particles can cause manufacturing defects. Also, small changes in the process might lead to a small variation somewhere on the wafer so that the die at that location will not function properly anymore. New process technologies usually exhibit a subpar yield that will increase over time while the chip manufacturer fine tunes the process and its design rules. Mature nodes that have been in production for years usually perform very well and a large number of dies per wafer will work as expected.

Design for test (DFT) is sometimes an underestimated aspect in the design process; however, designing a chip without test structures can be compared to designing RTL without a thorough verification strategy. In both cases the chances of getting a fully functional chip are slim.

The International Technology Roadmap for Semiconductors [93] sees test cost as a key challenge in the near future. Therefore the correct test approach is also determined from an economic viewpoint.

The following chapter will outline the testing methodology that has been implemented for the EXTOLL chip and work out an approach to perform testing in-house before final PCB assembly.

### 5.1 Test Analysis

The following figure 5.1 shows the available options that can be considered to test produced ASICs after wafer production.

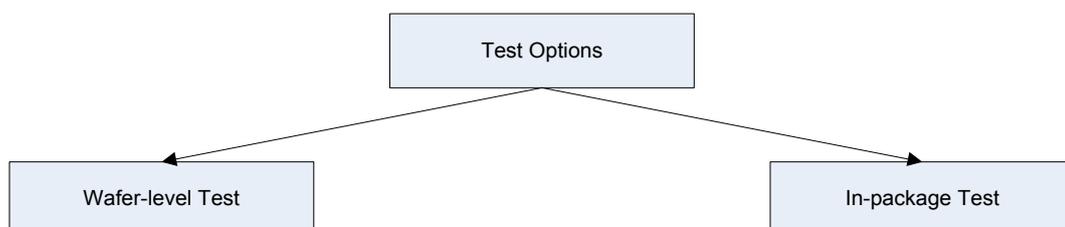


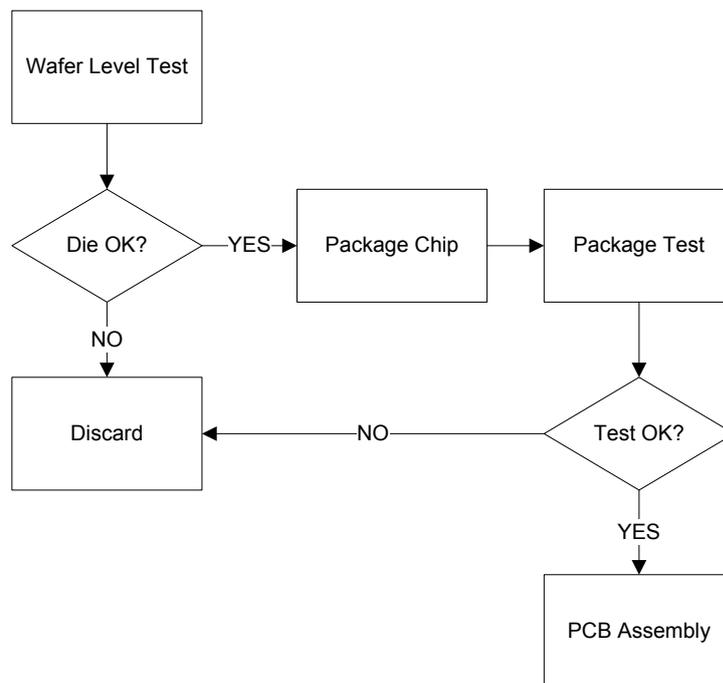
Figure 5.1 Test Options

## EXTOLL Test

---

Usually, both test options are taken into account. The first test is performed at wafer level before the individual dies are sliced. At this point, defective dies are marked as bad (in former days with an ink spot, hence the name inking, nowadays this is done by a digital inking process) and sorted out after the wafer is scribed and cut so that only known good dies will be packaged by the packaging house. The goal of this test is to actively sort out devices with major defects so that they will not be packaged. Thus, the packaging cost can be saved for these dies.

In general, a second test is also performed after packaging which can also detect problems that were introduced during this step, but a more thorough test can be run on the final chip, too. Since wafer tests are not as thorough as they should be to find all defects a more comprehensive set of test patterns are run on the packaged chip. Afterwards it is guaranteed that only tested chips that are fully operational (in relation to the tests performed) will be shipped to the PCB assembly house. This common flow is shown in the following figure 5.2.



**Figure 5.2 Chip Test Flow**

The following two paragraphs give a short overview of the two test methods before the cost is analyzed for each of them. Afterwards the DFT features for EXTOLL are shortly presented including the main problems during test before a testing platform is introduced.

## 5.2 Wafer Test

Wafer test needs special Automatic Test Equipment (ATE). This equipment consists of the tester itself which runs the test program that controls the actual tests, the testing logic and the wafer stepper. In order to connect to the bare dies on the wafer an adapter is needed, the so called probe card. The following two figures 5.3 and 5.4 show an example of current state-of-the-art equipment needed for high-performance chips. It is clear that the tester must be more complex if the chip to be tested contains more logic or advanced features.



Figure 5.3 Teradyne Tester<sup>5</sup>

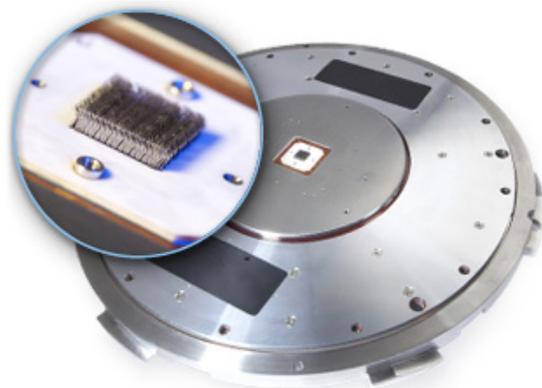


Figure 5.4 Microprobe Vx-MP Probe Card<sup>6</sup>

While the tester itself is already available at the test house both the probe card and the test program must be developed specifically for the device under test. Several factors limit the testing process:

- The tester only has limited memory to store patterns and their responses. For complex designs these patterns can be rather memory consuming and the more logic the chip contains the more patterns must be analyzed to receive good test coverage.
- Power consumption is a big issue during test which will also be discussed later in the chapter. A typical C4 bump can deliver about 50mA of current to the device which means that a large number of bumps must be connected to supply adequate power for test operation. During touchdown each probe pin applies a force of 5-10 g on the die [94] so that the number of probe pins is also limited due to mechanical reasons.

<sup>5</sup> <http://www.teradyne.com>

<sup>6</sup> <http://www.microprobe.com>

As a conclusion this means that thorough tests will not be possible during wafer test and only basic defects will be detected during this step.

- The testers lack functionality to test high-speed structures like multi-gigabit transceivers that are more and more common in modern SoC design.
- In general, high-speed signals are problematic. Most testers are not able to supply clocks running at several hundred MHz.
- The probe card itself is only good for a limited number of touchdowns. With each tested die the probe pins will be more blunted until they are no longer sharp enough to connect the pad structures on the wafer reliably. Since the pins are also built with  $\mu\text{m}$  structures they can also break due to mechanical stress.
- The tester steps over the complete wafer and must test all dies. The more complex the test program is the more time this process consumes. However, tester time is not cheap and every second spent on wafer test costs money.

### 5.3 Package Test

Because of the limitations outlined in the previous paragraph wafer level testing is not as thorough as it should be to find all defects in the chip. Therefore it is essential to run a second phase of tests with more exhaustive coverage after the dies have been packaged. This process will sort out again a number of defective chips so that only fully tested chips are prepared for final assembly or shipment to the customer.

In this phase chips are already packaged and balled. This means that the tester needs some kind of adapter to connect to these balled BGAs. There are three kinds of connection structures for this application [95]:

- Pogo pins
- Elastomer connectors
- MEMS cantilever probes

The most common structure is the pogo pin which is also called spring pin or spring probe pin and is shown in figure 5.5<sup>7</sup>.



**Figure 5.5 Spring Pin**

---

<sup>7</sup> <http://www.ironwoodelectronics.com>

As with all test structures physical endurance is a major concern. A test socket must work reliably for at least several thousand test cycles. At the same time it should avoid damaging the balls on the chip. Since package tests should also support tests in the multi-gigabit range the electrical characteristics of the connection should not have a large influence on the signal integrity.

## 5.4 Process Analysis

Every additional step in ASIC production means additional costs which leads to either a decrease of profit margins or higher retail prices. In order to define the cost of a chip all parameters must be analyzed to see whether there is room for improvement somewhere. Because each project has its own unique requirement there is no single solution that is fitting for all purposes. In the following a cost model for the EXTOLL ASIC will be developed. A generalized statement by Gordon Bell says that when the volume doubles, costs reduce by 10% [96]. But predicting the actual cost of an ASIC including test is complex and depends on many variables. However, available models [97] are sometimes too detailed when a more general point of view could also give enough insight without taking into account dozens of parameters.

In general, costs can be divided into NRE costs and unit costs as discussed in chapter 2. The NRE costs are always fixed for a stage in the design process. In the following the costs per unit will be laid out. At first some naming conventions have to be introduced in the following table 5.1:

Variable	Definition
<b>GDW</b>	Gross dies per wafer. This is the number of dies that can be cut out of a wafer
<b>D<sub>w</sub></b>	The number of dies that will be tested during wafer test
<b>W</b>	The number of wafers produced
<b>D<sub>p</sub></b>	The number of dies that have been identified as not defective during wafer test. These dies will be packaged
<b>Y<sub>WT</sub></b>	The yield achieved during wafer test
<b>Y<sub>PT</sub></b>	The yield achieved during package test
<b>KGD</b>	Known good dies. This is the number of dies per wafer that are fully tested and operational.
<b>Y</b>	Overall yield

Table 5.1 Parameters for Die Calculation

The gross number of dies gained from a single wafer can be calculated as follows [98]:

$$GDW = \left( \frac{\pi R_{eff}^2}{A} \right) e^{-\frac{0.58(H+W)}{R_{eff}}}$$

$R_{eff}$  denotes the effective radius of the wafer, i.e. the physical radius without the edge exclusion.  $A$  is the die area including the scribe lane. For a rectangular chip  $H$  and  $W$  signify height and width of the die, for a quadratic chip  $H$  and  $W$  are identical, of course.

The number of dies gained from  $W$  wafers is then

$$D_W = GDW \cdot W$$

A number of these dies will be sorted out during wafer test. Depending on the yield the number of dies to be packaged is

$$D_P = D_W \cdot Y_{WT} \Leftrightarrow D_W = \frac{D_P}{Y_{WT}}$$

Once again, a certain number of dies will be sorted out during package test so that the number of known good dies  $KGD$  is defined as

$$KGD = D_P \cdot Y_{PT} \Leftrightarrow D_P = \frac{KGD}{Y_{PT}}$$

The total yield of a wafer is then described as the percentage of good dies from all dies that were sliced.

$$Y = \frac{KGD}{GDW}$$

Alternatively the overall yield  $Y$  can be calculated as the combination of the yield number of the two test phases.

$$Y = Y_{WT} \cdot Y_{PT}$$

From these numbers the cost of running test can be determined. For simplicity reasons it is assumed that a defective chip is tested as long as a good die (and thus causes the same costs) although normally the test would be aborted as soon as a defect was found.

Since the cost to test a die is now time invariant only a few new metrics have to be introduced in the following table 5.2:

Variable	Definition
$C_{WT}$	The cost of testing a single die during wafer test
$C_{PT}$	The cost of testing a single die during package test
$C_P$	The cost of packaging a single die
$NRE_{WT}$	NRE costs of wafer test
$NRE_{PT}$	NRE costs of package test
$NRE_P$	NRE costs of package development
$TC_{WT}$	Total cost of wafer test
$TC_{PT}$	Total cost of package test
$TC_P$	Total cost of packaging

Table 5.2 Parameters for Cost Calculation

The total cost of wafer testing can be calculated as follows:

$$TC_{WT} = D_W \cdot C_{WT} + NRE_{WT}$$

Essentially, for a wafer level test the cost can be broken down to the setup cost of the tester and the time the wafer stays in the machine to perform the tests.

The cost of packaging is determined in a similar way:

$$TC_P = D_P \cdot C_P + NRE_P$$

At last, in order to test these packaged chips the cost is defined as:

$$TC_{PT} = D_P \cdot C_{PT} + NRE_{PT}$$

In total the cost of test  $TC$  is defined as the sum of these three components:

$$TC = TC_{WT} + TC_P + TC_{PT} = D_W \cdot C_{WT} + D_W \cdot Y_{WT}(C_P + C_{PT}) + \sum NRE$$

Besides the total cost it is interesting to see how much money must be spent to get a specific number of good dies. This cost per unit  $C$  can be calculated by considering the yield of both test runs:

$$C = \frac{TC}{Y_{WT} \cdot Y_{PT} \cdot GDW \cdot W}$$

During package test the same tests that are performed on the wafer can also be run. Thus, it is a reasonable suggestion to skip wafer test completely and concentrate on package test. In order to determine the best test strategy one should compare the costs associated with both

## EXTOLL Test

---

options. In order to do this the costs should be calculated as function of known good dies since this is the desired result in the end.

$$f(KGD) = \frac{KGD}{Y_{PT}}(C_{PT} + C_P) + \frac{KGD}{Y_{PT}Y_{WT}}C_{WT} + NRE_{WT} + NRE_{PT} + NRE_P$$

The above function calculates the cost for a number of working dies while taking into account that two steps of tests are performed, once on wafer level and once in the package.

$$g(KGD) = \frac{KGD}{Y_{PT}}(C_{PT} + C_P) + NRE_{PT} + NRE_P$$

The second function also calculates the cost for a number of working dies, however, in this case only package tests are performed. It is important to note that the yield number  $Y_{PT}$  in the two functions is not the same. In the second function  $g(KGD)$   $Y_{PT}$  also includes dies that were sorted out in the first function during wafer test and is therefore significantly lower.

The NRE costs for setting up both packaging and the package test accrue in both cases and can therefore be neglected for the analysis. The following figure 5.6 plots the cost development for both variants for different yield numbers. The NRE costs for wafer test are assumed to be in the range of 100,000\$.

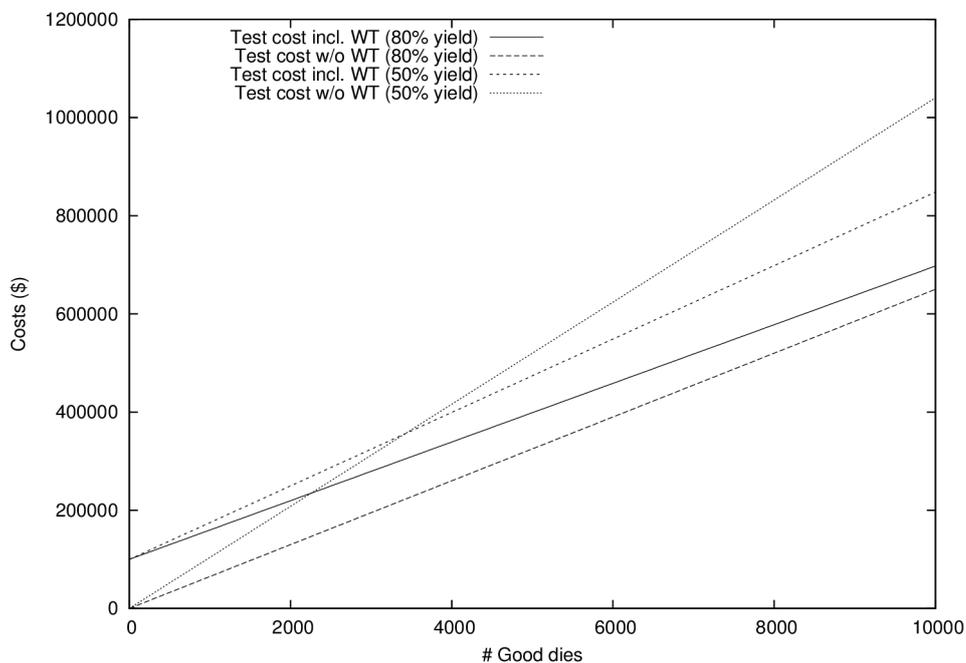


Figure 5.6 Test Cost Analysis

The crossing point of two correlated plots defines the point after which setting up wafer test would be preferable in order to save cost. However, it should be made clear that the numbers inserted into the formulas are based on rough estimates gained from publicly available resources and might not be completely accurate for the EXTOLL project.

A clear conclusion on the best test approach cannot be given since it all comes down to the yield numbers. In the case of a good yield wafer tests might be a waste of time and money since almost all of the chips will be packaged. On the other hand, a wafer test will already sort out lots of defective chips in the case of sub-par yield so that lots of packaging costs can be saved. Therefore it is prudent to completely package the first batch of (prototype) chips in order to determine approximate overall yield numbers so that subsequent production runs can be optimized. It is clearly visible from the plot above that for a bad yield the crossing point is reached already at small numbers and for a volumes of about 10,000 chips savings in the range of several 100,000\$ can be achieved while the crossing point for an 80% yield is not even reached in the plot.

In the case of EXTOLL the minimum order for the first run will already produce enough chips for the entire lifetime of the product even with a conservative yield for a mature technology. Thus, wafer tests will be skipped for EXTOLL and all dies will be packaged anyway. However, if EXTOLL's successor enters the market with a higher volume this conclusion might be void and future products will be in need of wafer tests.

## 5.5 Test Setup

In order to develop the optimal testing methodology for EXTOLL one has to look at the available test options. Several DFT [99] features were included in EXTOLL that will be shown in the following section:

- **Internal Scan:** Flip-flops in the design have been replaced with so called Scan FFs that include a multiplexer in the datapath that switches between the normal data input and a scan input. These flip-flops are then connected ("stitched") to scan chains with each scan input being driven by the data output of the previous flip-flop as seen in the following figure 5.7.

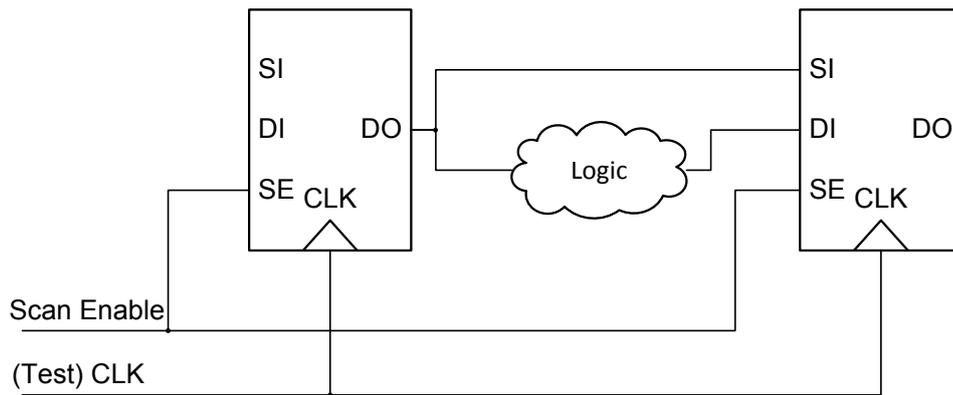


Figure 5.7 Scan Chain Structure

In order to determine the chip's correct functionality the following steps are performed by the tester:

- A global signal (scan enable, SE) switches all multiplexers of the scan flip-flops to the scan input
- A bit pattern is shifted in the chain
- The multiplexers are switched back to the regular data input by deasserting the scan enable signal
- One clock cycle is triggered so that the logic between two flip-flops is executed
- The resulting pattern is shifted out again
- If the results match the pre-calculated response no defect was found

Certainly, this process can overlap two consecutive tests. The cycles in which a result is shifted out can also be used to shift in the next pattern.

Of course, a single pattern will not be able to find all defects (e.g. a stuck-at 1 fault cannot be found if the response of the circuit would also produce a 1), therefore the ATPG tool will calculate patterns that will cover most of the error space.

The solution space for the ATPG tool is enormous because not all data inputs of flip-flops in the real design are dependent on the state of one other flip-flop like shown in the figure above, but rather on a combination of flip-flops. In order to find all defects in chains of tens of thousands of flip-flops many test patterns have to be calculated.

Due to the size of these patterns and to reduce testing time the patterns are loaded as compressed data and later decompressed on chip. The same process in reverse order will happen during shift-out.

- **Memory BIST:** EXTOLL contains more than 1,000 memory instances that are susceptible to manufacturing defects due to their structure. All memories include an internal self-test that can be controlled via JTAG. For accessing the test structures the JTAG TAP controller converts the JTAG commands to an IEEE 1500 compliant [100] access which is used to connect to the test master. The tester runs several march tests designed to detect the most common faults in memory so that it can be determined whether only correctable single bit errors exist or if the device cannot be saved.
- **Serializer BIST:** The serializers are also equipped with a built-in self-test that is controlled via JTAG, too. For testing, the serializer can be setup either with an internal loopback to test the analog structures or with external loopback to test for packaging problems. The test patterns are generated with a PRBS module that drives the transmitter and generates random traffic which is then sampled on the receiving side to check for consistency errors.

The following figure 5.8 gives a simplified overview of the internal features. Unfortunately, scan tests and the JTAG controlled testing functionality can not be controlled in parallel because the BIST logic is also scanable. This means that the state of the BIST logic is in an undefined state during scan test.

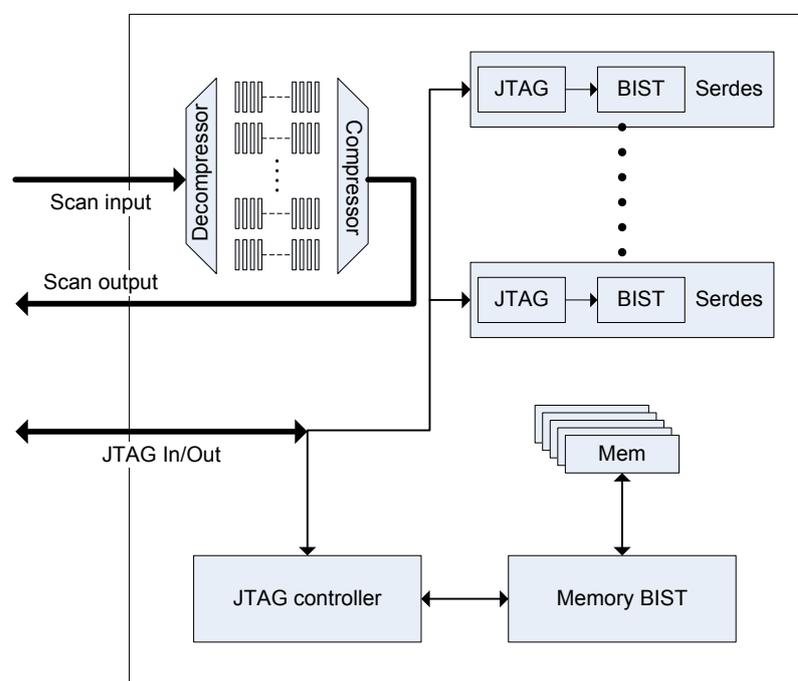


Figure 5.8 EXTOLL Test Organization

Besides the already mentioned global test features EXTOLL also supports On-Product Clock Generation (OPCG) to run at-speed tests without relying on an externally induced high-speed clock [101]. Instead the internal PLL is used to generate the operating frequency of the circuit. Test structures inserted in the clock path take care that only a single clock pulse is propagated through the design during at-speed tests.

Unfortunately, there is no way to test the integrated HyperTransport PHY because its documentation did not allow any insight to the offered testability options although scan in- and outputs are available. This means that using HyperTransport is a huge gamble because correct functionality is uncertain until the chip is assembled on an add-on card and is tested in a server system.

## 5.6 Test Hardware

Since the EXTOLL dies are not tested on the wafer and will be packaged regardless of their functionality there is need for an on-site tester for bring-up that will be able to perform all the aforementioned tests and – if a test fails - give feedback on the location of the defect so that an analysis can be developed whether there are yield sensitive areas in the design.

The following paragraphs analyze the requirements for such a platform and propose a system to perform those tasks in a cost-efficient way.

### 5.6.1 Analysis

In order to build a test system for packaged EXTOLL chips several aspects have to be taken into consideration:

- What are the important features that must be tested?
- How large is the power consumption during test?
- How does the clocking work?
- How are the tests performed and evaluated?
- How can the system be built as cheap as possible?

Testing is always a tradeoff between time and coverage. Of course, best test coverage can be achieved by running test patterns for a very long time; however, the test throughput suffers greatly. On the other hand, a short tester time will hardly be sufficient to perform enough tests to feel confident about the tested chip.

A test platform should be able to perform the following tests that are all included in EXTOLL as described in the previous section 5.4:

- Thorough scan tests both with test clock and also with OPCG
- Memory BIST
- Serializer BIST both with internal and external loopback

Since the tester will not be a production system it makes sense to also use it for characterization to see how well the chip behaves across several voltage corners. This means it should also support so called burn-in tests to accelerate aging effects like electro migration e.g. by applying higher voltages.

One of the biggest problems for testers is power consumption [102]. This is not only the case for wafer test where the restricted number of connectable probe pins limits the power that can be delivered to the DUT but also the generally higher power consumption while the chip is running in test mode. This is derived from the way power is consumed in the device.

Power consumption can be divided into three components:

- Static power is consumed even when there is no switching activity. This is caused by leaking effects, predominantly by gate leakage. However, for future technology nodes sub-threshold leakage is assumed to take over the dominant part [103]. Regardless, static power is irrelevant while looking at power consumption during test since it is consumed independently from switching activity.
- Dynamic dissipation because of shorts during switching. Because of the internal structure of CMOS logic there are always two parts in a logic gate that perform complementary operation: either the PMOS part conducts and the NMOS part does not conduct or the other way round. The PMOS part is responsible for generating a logic 1 on the output of circuit since it is connected to  $V_{DD}$  while the NMOS part is connected to ground. However, switching does not happen instantly. Instead both transistor types conduct at the same time for a finite period and there is a short between  $V_{DD}$  and ground.
- The most important case for test is the dynamic power consumption which is caused by switching activity since it is the largest contributor to the power requirements during test.

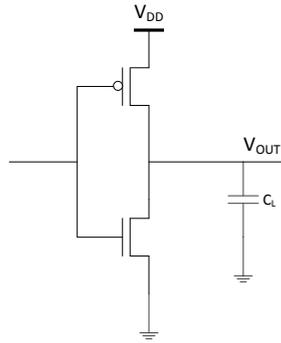


Figure 5.9 CMOS Inverter

The dynamic power consumption of a circuit can be derived best from a simple CMOS inverter as seen in figure 5.9. It consists of a p-channel and an n-channel MOSFET. If a '0' is applied to the inverter input the PMOS transistor conducts and a '1' appears at the output. In the reverse case, if a '1' is applied the NMOS transistor conducts and a '0' appears at the output. The output of the inverter is driving a capacitive load  $C_L$ .

During the transition from low to high energy is taken from the supply that can be calculated as follows [104]:

$$E_{VDD} = \int_0^{\infty} i_{VDD}(t)V_{DD}dt = V_{DD} \int_0^{\infty} C_L \frac{dv_{out}}{dt} dt = C_L V_{DD} \int_0^{V_{DD}} dv_{out} = C_L V_{DD}^2$$

The energy stored on the capacitor can be derived as:

$$E_C = \int_0^{\infty} i_{VDD}(t)v_{out}dt = \int_0^{\infty} C_L \frac{dv_{out}}{dt} v_{out} dt = C_L \int_0^{V_{DD}} v_{out} dv_{out} = \frac{C_L V_{DD}^2}{2}$$

The missing energy is dissipated in the PMOS transistor. During the high-to-low transition the energy on the capacitor is discharged and dissipated in the NMOS transistor so that the total energy is equal to  $C_L V_{DD}^2$ . The dynamic power consumption now depends on the number of times the load capacitance is charged and discharged. Since this directly relates to the frequency of the circuit the following formula applies:

$$P_{dyn} = C_L V_{DD}^2 f$$

During normal operation only a small part of the logic is actively switching so that the formula above can be enhanced with a corrective factor  $\alpha$ . During test, however, a much larger portion of the design is active because low power structures like clock gating are

disabled, all portions in the design are tested in parallel (e.g. in EXTOLL during normal operation the HT3 core and the PCIe core will never be active simultaneously) and the test patterns try to switch as many circuits as possible in as little time as possible. All this causes higher power dissipation. While this problem is alleviated during normal test operation because of the lower test clock frequency it affects at-speed tests heavily. ATPG tools can account for this issue by calculating test patterns that limit the switching activity in the chip to a certain amount [105]. In turn this means that more patterns must be generated and analyzed to generate the same coverage which increases the overall time on the tester.

A test platform should be able to drive all clocking pins that exist in EXTOLL. This means, external components like an oscillator, a high-speed backup clock must be included. Of course, it is also advantageous if the possibility to introduce a clean reference clocks from an external clock generator is added.

EXTOLL features several configuration and reset pins that are described in table 4.7. These pins should not be static but configurable from the tester so that the chip can be put into different modes.

The patterns that are generated by the ATPG tool consume lots of space, not only because they must contain large test vectors, but also because they are saved in a text based and therefore also human-readable format, the Standard Test Interface Language (STIL) [106].

The test platform must be able to load this data from an external storage, shift the patterns into the chip and in return also analyze the results and compare them with the expected response.

The last point is not technological, but nevertheless crucial. The tester cost should not exceed certain boundaries. Since it is not mass produced there will not be any discounts for the components based on high volumes. Even if components are shared between the tester and the final EXTOLL PCB the synergy effects are little because tester assembly will predate the EXTOLL PCB production and it does not make sense to order large quantities of parts weeks or months in advance. The key message is that the tester must be as cheap as possible while offering as much features as possible, a cost of tens of thousands dollars will greatly reduce the economic viability.

### 5.6.2 Proposal

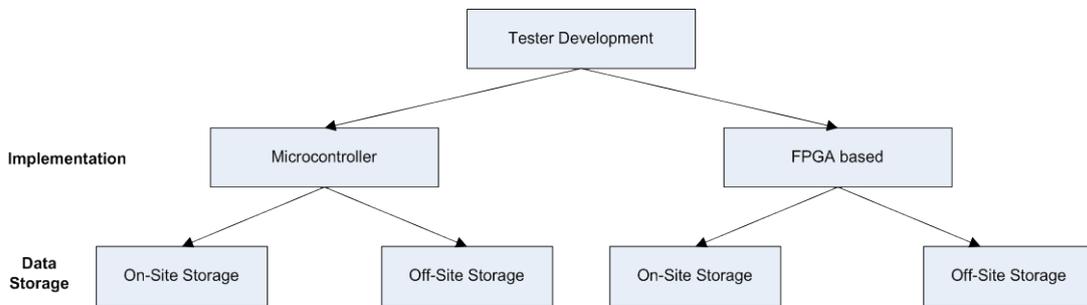
As mentioned before test patterns are saved in a text based format. An example of such a STIL file is given in appendix D. While the format facilitates readability for humans it is not optimal for automatic processing. Thus, it is usually only used as an intermediate format that is translated to a native format understandable by the ATE by an interpreter program developed by the vendor of the test equipment. This flow is shown in the following figure 5.10:



**Figure 5.10 Test Data Processing**

A similar approach is also necessary for the proposed tester since processing the STIL format directly on the tester would require both a fast CPU and also lots of RAM resources.

The tester development consists of two design decision levels as seen in the following representation 5.11: the technology that will be used to implement the design and the question whether test data is stored directly on the tester or not.



**Figure 5.11 Design Space for Tester Development**

Test data must be made available to the tester. Two approaches can be considered for this: on-site and off-site storage. On-site storage would require access to mass storage by the test platform, either via SATA or a simpler SDIO interface for flash based memory cards. While it makes the tester independent of external devices it also means that the data must be replaced in case of an updated test procedure. Also, feedback of the test progress and failed

patterns might be less detailed or can only be analyzed afterwards. At last although the concern might be minor nowadays, test patterns are limited by available disk sizes.

Off-site storage in return requires an external host that continuously pushes test data to the tester. This means that an interface must be available on the tester that is able to handle this amount of data. In this case, the user of the tester gets almost immediate feedback if a test fails and can react accordingly via the host PC that controls the tester. Depending on the complexity and usability of the tester program progress and problems can be identified more accurately.

Looking at the two options it can be said that both approaches will work fine, however, the additional features that come with off-site storage make this option the preferable one.

The hardware implementation of the device that controls the tester also shows two paths: a microcontroller that executes a specific test program or an FPGA that contains a tester program in hardware.

Microcontrollers are not only available as small and slow devices, but also with powerful processing cores and lots of versatility. As an example, the F4  $\mu$ -controller series by ST Microelectronics [107] includes an ARM Cortex M4 core running at up to 168 MHz, lots of integrated communication interfaces like SDIO, USB2, I<sup>2</sup>C and Ethernet as well as up to 136 GPIO pins for about 10\$ per unit<sup>8</sup>. This means it could perform scan tests by reading the data from an attached mass storage device, communicate with the ASIC's scan interface through its GPIO pins and then write back the results to the mass storage. However, because of the integrated interfaces the device is also well equipped to handle off-site storage of scan patterns that could be transferred by USB, for example. Since microcontrollers nowadays are usually programmed in a high-level language instead of assembler development of a test module should not be more challenging as any other software project. One problem of the microcontroller is the fact that the GPIO pins cannot be run at arbitrary speeds but is limited to a selection of predefined values.

An FPGA in its basic state only consists of a bunch of pins without any intelligence. This means a test design must be completely developed from scratch in Verilog which might be a little bit harder than developing in C for example. Mass storage access would be hard since the complete functionality including protocol stack had to be developed in hardware. However, an FPGA can easily connect to other interfaces like an Ethernet or USB port so

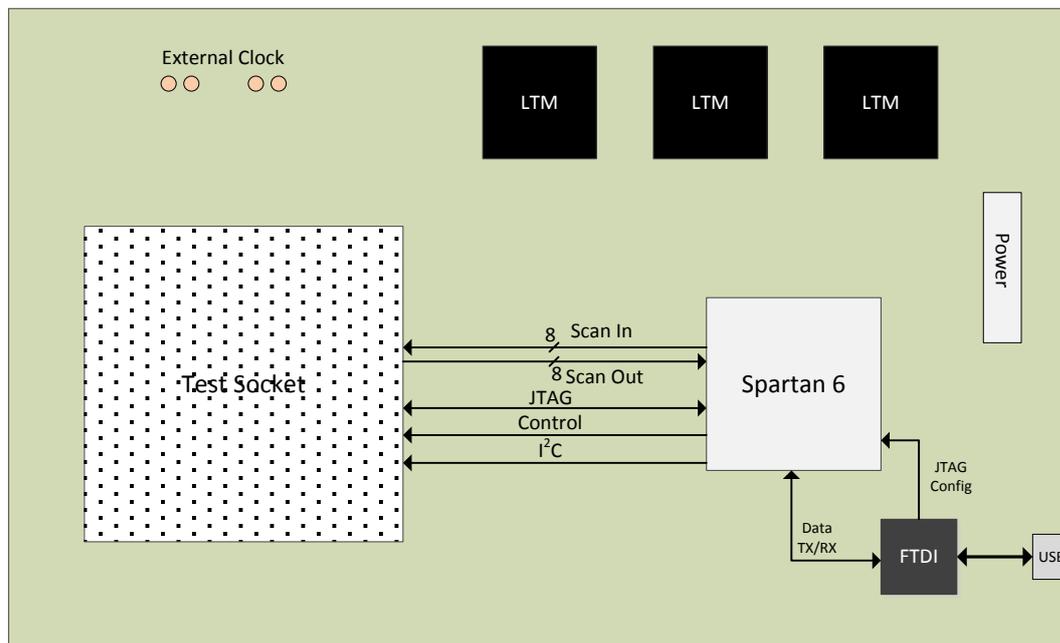
---

<sup>8</sup> Price from November 2012, [www.digikey.com](http://www.digikey.com)

that off-site storage as the preferred solution is no problem. One of the biggest benefits of using an FPGA is the fact that the timing of pins can be controlled accurately. This means not only that the speed of the tester can be tuned in an acute range but also that it is easy to compensate for I/O timing problems. Unfortunately, a simple FPGA will already cost more than a microcontroller chip.

Although a microcontroller seems to be a good solution the proposal for the tester includes an FPGA based test controller. This is mainly based on the fact that there is already lots of expertise for developing hardware. Since a design for the tester must be developed anyway it makes sense to do this development in a familiar environment. Verification for a hardware solution is also easier since verification components are already available. Of course, it is also possible to verify the tester together with the EXTOLL design in a joint simulation beforehand. Essentially, the higher cost of an FPGA solution is negligible in relation to the cost of the ASIC.

A schematic representation of the proposed platform is shown in the following figure 5.12:



**Figure 5.12 Test Platform Proposal**

It contains a low-cost FPGA (e.g. Spartan6 by Xilinx) and a USB2 interface chip by FTDI [108] that constitute the main functionality of the tester. The FPGA is responsible for shifting in and out scan patterns to the EXTOLL chip under test and also controls its JTAG interface.

The USB chip contains two channels and connects via USB2 to the host computer. One of the channels is configured as a JTAG controller and controls the JTAG port of the FPGA to load the design into the device. The other channel is configured as a FIFO interface and is responsible for controlling the operation of the tester and the data exchange of scan patterns and results between the host computer and the test platform.

The packaged EXTOLL chip is inserted into a socket that is soldered onto the tester's PCB. Its scan inputs and outputs as well as the JTAG and control pins are connected to the FPGA. The PCB stackup can be relatively simple and the production therefore low priced since no high-speed interfaces will be routed on the board. The serializer outputs will directly be connected to the corresponding receive channel so that external loopback tests can be run.

At last, the test board only contains DC/DC converters and LDOs for all the power supplies needed by EXTOLL. For higher flexibility, power can also be injected externally. This is also the case for all clocks that are needed to run the tester. In order to debug and measure as much data as possible the board should contain many measurement points where a scope probe can be attached.

Because of the higher power consumption during test a cooling solution must also be installed. It remains to be seen whether a passive heatsink with external fans blowing air might suffice or if an active cooling solution has to be installed.

In the following a short cost estimate is given for the tester (all numbers for small quantities)<sup>9</sup>:

- A Spartan6 FPGA is available for about 50\$ with 45k LUTs which should be enough logic elements to realize the required functionality.
- The FTDI USB chip is available for 5\$.
- The socket adapter is the most expensive component on the board. A first estimate lies in the range of 1000\$ per socket.
- The power supply is handled by power modules by Linear Technology that were already tested on the prototype boards. These modules are available for about 30\$ each.
- For the PCB a simple 6 layer stackup might be sufficient. Stackup and size requirements are similar to the PCIe backplane presented in paragraph 2.7 so that a price of about 100\$ per PCB is possible.

---

<sup>9</sup> All prices looked up in December '12

## EXTOLL Test

---

- Assembly for a board with few BGA devices can be assumed to be less than 500\$ per board.
- For the remaining components (including SMD capacitors and resistors) an additional 200-300\$ should suffice.

Altogether, the complete tester should be not more expensive than about 2000\$ for small quantities. As with every hardware projects production costs decrease rapidly when quantities go up. However, for the tester anything higher than 10 would be a large discrepancy with regards to the expected volume of EXTOLL chips.

With this test platform there is an easy and cost efficient way to test packaged EXTOLL dies. Analysis of the testing process and the associated problems showed that a package test is not only able to find defects that would be identified during wafer-test but can also run more detailed tests because it can deliver more power to the chip. Furthermore testing of serializer structures with external loopback is only possible during the final test phase. With regards to the cost analysis it can also be cost efficient to skip wafer tests altogether in cases of a high yield and a low volume.

## 6 Conclusion

### 6.1 Results

The goal of this thesis was to look at the methodology and the surrounding ecosystem for the design of a complex network ASIC. It was analyzed that it is not enough to only look at the optimization of a single component in an ecosystem. Instead, due to the high integration everything is interwoven, and the optimization space suddenly becomes multi-dimensional and all processes must be analyzed and improved concurrently.

In this context four different aspects in the design process were analyzed in this thesis and solutions or improvements are presented for each of them.

By looking at the competitors and analyzing the technological options it was determined that EXTOLL must be implemented as an ASIC to deliver the highest possible performance. Due to technological differences designs cannot be directly ported to another technology although they are written in the Verilog HDL. One of the key differences is the handling of memory blocks. ASIC memories are susceptible to bit errors. The probability for a bit error increases with the number of RAM instances and the area they occupy on the die. Thus, it is essential that error checking and correction is implemented in all memories. This work introduces a memory generator script that can build memories of arbitrary width and depth automatically based on a small set of building blocks. These memories are internally protected with ECC logic while keeping this functionality hidden from the user. This means that these generated memories can be plugged into the design without rewriting any code and sacrificing performance. The script is used to generate several dozen of diverse memory configurations that are used in the EXTOLL ASIC and generates more than 20,000 lines of Verilog code. It has become an efficient and integral part of the design process.

The ASIC design flow is both complex and time consuming. A scripting based approach in combination with a version management system and a concise data organization can enable collaborative work to reduce the overall time needed to finish the project. As backend design is an iterative approach it makes sense to work with a basic set of parameters that define the design and all further design requirements are derived from them. For example, if a basic condition like the size of the die must be changed all following steps have to be updated. For

## Conclusion

---

example, the number of bumps has to be changed, the placement of the large serializer blocks must be adapted, the floorplan must be redefined and many more areas will also be affected. Special consideration in the design flow was given to the floorplanning process as it is one of the key elements for a successful (which means both routable and able to achieve timing closure) implementation. A floorplanning methodology was developed that takes information from the package development into account and also depends on a datapath analysis of the design.

Until the signal of a high-speed link arrives at the connector it has crossed several hierarchies in the signal path. Each of these hierarchies comes with unique requirements that narrow the design space. In this thesis each level was analyzed and in turn a package definition was proposed that is seen as an optimal solution considering the technological limitations, the viability, signal integrity and also cost concerns.

The testing of dies after the wafers have been processed in the fab is a cost-intensive task which might not be efficient for small volumes. In order to determine the best test approach a cost model is developed. Furthermore the DFT structure in EXTOLL are analyzed and the problems regarding power consumption during test are considered for both wafer-level and package based testing. In order to run full tests to qualify chips in a lab environment this thesis contributes a proposal for an integrated FPGA based test platform that is able to run all test features thoroughly.

After going through the complete design process and significantly improving several aspects of it a high confidence could be reached that EXTOLL will be optimized as good as possible and the solutions worked out in the thesis helped to advance the overall design methodology.

Despite many challenges in the overall development it was possible to develop a highly complex network chip with a low budget and a small team of highly engaged researchers. The chip is currently scheduled for tapeout in Q1/2013 so that market introduction can happen in the 2<sup>nd</sup> half of the same year which will be in time for the DEEP prototype to show its capability to scale to the exascale era. However, there are also several other markets to explore and many collaboration possibilities with researchers and companies from all over the world are expected.

## 6.2 Project Review

It can be said that the development of a highly complex chip like EXTOLL was not an easy task. In retrospect it must be stated that the team of researchers was too small so that inevitable delays had to happen in the design process. But not only can the high workload be blamed for the repeated shift of the tapeout date but also problems with acquiring external IP components and the funding of the project.

However, some lessons could be learned from this venture that might be able to speed up further designs and avoid repeating mistakes that were made before.

- FPGAs should only be used for verification and prototyping purposes. The task of developing and maintaining a separate codebase that is usually necessary because of performance issues is wasting resources that are missing somewhere else in the design process. The verification effort increases as well. Although the development of the Ventoux board showed that an FPGA can make a competitive product, its price/performance ratio is worse than a high-end ASIC implementation. It makes more sense to build a small number of prototyping boards that can emulate most of the design's functionality. Developing two or more products in parallel should only happen if there are enough engineers available so that all projects can be worked on with sufficient resources.
- Verification is supposed to consume 70% of the overall design effort for an SoC [109]. A study [110] commissioned by Mentor Graphics showed that on average almost eight verification engineers worked on a design with 1 to 20 million gates which is consistent with the complexity of the EXTOLL design. Thus, verification should already start as early as possible during the design phase and as many resources as possible should be assigned to this task. The more complex the design becomes the harder it is to be sure that all bugs are found and fixed. It is always more expensive to find bugs later in the end phase of the implementation or even in the final chip.
- Design guidelines that were introduced in chapter 2 make sure that the developed RTL is already as portable as possible. Careless design just because something works on an FPGA means that the code must be rewritten again later in the design process. A linter [111] can detect common design mistakes early on and prompt the designer to fix them.

## Conclusion

---

- Depending on the timeframe it should be carefully analyzed if it is possible to develop all necessary IP in-house or in cooperation with another university. Depending on external IP providers can unexpectedly delay or even jeopardize the whole project because offered IP is no longer available or is downgraded in performance so that it does not meet the expectations anymore. Within a timeframe of 1-2 years it should be possible to develop a SERDES, for instance, with a team of two or three analog experts and two MPW test runs at a cost that will be less than the amount of money that is charged by commercial vendors for an IP license. However, since standard cell libraries and sometimes memories are either offered for free by the foundry or can be acquired with a comparably small price tag it probably will not be profitable to shift these developments in-house.
- A clear and concise feature list should be designed early in the project. Some modules in EXTOLL were rewritten several times to accommodate new features that were originally not planned for the design. This wasted not only time for the engineer that had to write the code but also hindered the verification process.
- Bug tracking software like Trac [112] should already be used from the project start and continuously be updated. It gives an excellent overview of open tasks for a specific design milestone and tracks the progress that has already been made on the project. It tightly integrates with the version management system so that changes that were made in the RTL code to fix a bug can directly be referred in the bug report.

## 6.3 Outlook

A lot of things could be learned from undertaking such a large project as the EXTOLL chip. While the first implementation was implemented on a relatively old and mature and therefore cheap process node the next implementation will have to advance technologically. A node change to a current 28nm process will allow further performance improvements not only regarding the internal frequency but also with serializers that support operation at rates higher than 10 Gb/s.

However, using a 28nm process will also mean a high investment. In order to minimize the risk of schedule changes IP development as suggested in the previous paragraph should start as early as possible. New features that will be included in EXTOLL have to be defined within the next months and both design and verification must commence as soon as possible if the planned schedule of a subsequent tapeout within 2 years should be reached.

Like the adoption of EXTOLL in the DEEP system in a specialized hardware there are considerations to build a bareblade system that removes all support chips like the Southbridge from the mainboard by integrating the functionality into EXTOLL. This would allow building a system with a low number of components that is highly dense and can be stacked efficiently to build large clusters of compute nodes.

Another research interest is the combination of a CPU with a network, namely EXTOLL. This integration could happen as a multi-chip-module in the first step, but the final goal is the integration of the interconnect's functionality in the CPU. While this will most likely not be possible with mainstream x86-based CPUs (except probably with a low performance Atom processor) there are several other CPU families that are more open.

In the end it can be said that EXTOLL's development is not over yet and will continue over the next years.



## A Acronyms

AC	Alternating Current
AMD	Advanced Micro Devices
ASIC	Application Specific Integrated Circuit
ATE	Automatic Test Equipment
ATOLL	Atomic Low Latency
ATPG	Automatic Test Pattern Generation
ATU	Address Translation Unit
BCH	Bose-Chaudhuri-Hocquenghem
BGA	Ball Grid Array
BI	Booster Interconnect
BIST	Built-In Self Test
BN	Booster Node
C4	Controlled Collapse Chip Connection
CDR	Clock-Data Recovery
CEO	Chief Executive Officer
CML	Current Mode Logic
CMOS	Complementary Metal-Oxide-Semiconductor
CN	Cluster Node
COTS	Commercial Of The Shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CTS	Clock Tree Synthesis
DC	Direct Current
DEC	Double Error Correction
DECTED	Double Error Correction, Triple Error Detection

## Acronyms

---

DEEP	Dynamic Exascale Entry Platform
DEF	Design Exchange Format
DFT	Design For Test
DRC	Design Rule Check
DSM	Deep Sub Micron
ECC	Error Checking and Correction
EDA	Electronic Design Automation
EP	Endpoint
EXTOLL	Extended ATOLL
FF	Flip-Flop
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
FU	Functional Unit
GPIO	General Purpose I/O
GUI	Graphical User Interface
HDL	Hardware Design Language
HPC	High Performance Computing
HT	HyperTransport
HTAX	HyperTransport Advanced Crossbar
I/O	Input / Output
I <sup>2</sup> C	Inter-Integrated Circuit
IB	InfiniBand
IBM	International Business Machines
IP	Intellectual Property
ITRS	International Technology Roadmap for Semiconductors
ITRS	International Technology Roadmap for Semiconductors
JEDEC	Joint Electron Devices Engineering Council

JTAG	Joint Test Action Group
LDO	Low-Dropout Linear Regulator
LEF	Library Exchange Format
LIB	Liberty Library File
LP	Link Port
LSB	Least Significant Bit
LVS	Layout Versus Schematic
MBIST	Memory Built-In Self Test
MEMS	Microelectromechanical Systems
MLM	Multi Layer Mask
MMU	Memory Management Unit
MPI	Message Passing Interface
MPW	Multi Project Wafer
MSB	Most Significant Bit
NIC	Network Interface Controller
NoC	Network-on-Chip
NP	Network Port
NRE	Non-Recurring Engineering
OPC	Optical Proximity Correction
OPCG	On-Product Clock Generation
PCB	Printed Circuit Board
PCIe	PCI Express
PDN	Power Distribution Network
PLE	Physical Layout Estimation
PLL	Phase-Locked Loop
PRBS	Pseudo Random Bit Stream
RAM	Random Access Memory
RDL	Redistribution Layer

## Acronyms

---

RF	Register File
RFS	Register File System
RMA	Remote Memory Access
ROI	Return Of Investment
RP	Root Port
RS	Reed-Solomon
RTL	Register Transfer Level
RX	Receive
SATA	Serial Advanced Technology Attachment
SDC	Synopsys Design Constraint
SDIO	Secure Digital Input Output
SECDED	Single Error Correction, Double Error Detection
SERDES	Serializer, Deserializer
SI	Signal Integrity
SMFU	Shared Memory Functional Unit
SNQ	System Notification Queue
SoC	System-on-Chip
SPI	Serial Peripheral Interface
SSC	Spread Spectrum Clocking
SSN	Simultaneous Switching Noise
STIL	Standard Test Interface Language
TCL	Tool Command Language
TSMC	Taiwan Semiconductor Manufacturing Company
TX	Transmit
UMC	United Microelectronics Corporation
USB	Universal Serial Bus
UVM	Universal Verification Methodology
VCT	Virtual Cut-Through
VELO	Virtualized Engine for Low Overhead

VIP	Verification IP
XBAR	Crossbar
XML	Extensible Markup Language



---

## B Bibliography

- [1] *TOP500 Supercomputing Sites*, <http://www.top500.org/>, [last accessed: 21-Sep-2012]
- [2] *Linpack Benchmark*, <http://www.netlib.org/benchmark/hpl/>, [last accessed: 21-Sep-2012]
- [3] P. Kogge, S. Lead, D. Campbell, J. Hiller, M. Richards, and A. Snavely, *ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems*
- [4] ITRS, *System Drivers*, in *International Technology Roadmap for Semiconductors*, 2011
- [5] M. J. S. Smith, *Application-Specific Integrated Circuits*. Prentice Hall, 2008, p. 1040, ISBN 0321602757
- [6] I. Kuon and J. Rose, *Measuring the Gap Between FPGAs and ASICs*, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, Feb. 2007
- [7] L. Rzymianowicz, *Designing Efficient Network Interfaces For System Area Networks*, Dissertation, University of Mannheim, 2002
- [8] HyperTransport Consortium, *HyperTransport<sup>TM</sup> I / O Link Specification*. 2008
- [9] B. Kalisch, A. Giese, H. Litz, and U. Brüning, *HyperTransport 3 Core: A Next Generation Host Interface with Extremely High Bandwidth*, in *1st International Workshop on HyperTransport Research and Applications*, 2009
- [10] A. L. S. Loke, B. A. Doyle, M. M. Oshima, W. L. Williams, R. G. Lewis, C. L. Wang, A. Hanpachern, K. M. Tucker, P. Gurunath, G. C. Asada, C. O. Lackey, T. T. Wee, and E. S. Fang, *Loopback architecture for wafer-level at-speed testing of embedded HyperTransport<sup>TM</sup> processor links*, in *2009 IEEE Custom Integrated Circuits Conference*, 2009, pp. 605–608
- [11] A. L. S. Loke, B. A. Doyle, S. K. Maheshwari, D. M. Fischette, C. L. Wang, T. T. Wee, and E. S. Fang, *An 8.0-Gb/s HyperTransport Transceiver for 32-nm SOI-CMOS Server Processors*, *IEEE Journal of Solid-State Circuits*, pp. 1–1, 2012

## Bibliography

---

- [12] PCI-SIG, *PCI Express Base Specification Revision 3.0*. 2010
- [13] T. Reubold, *Design, Implementation and Verification of a PCI Express to HyperTransport Protocol Bridge*, Diploma Thesis, University of Mannheim, 2008
- [14] B. Geib, *Hardware Support for Efficient Packet Processing*, Dissertation, University of Mannheim, 2012
- [15] H. Litz, *Improving the Scalability of High Performance Computer Systems*, Dissertation, University of Mannheim, 2010
- [16] H. Litz, H. Froening, M. Nuessle, and U. Bruening, *VELO: A Novel Communication Engine for Ultra-Low Latency Message Transfers*, in *2008 37th International Conference on Parallel Processing*, 2008, pp. 238–245
- [17] M. Nüssle, M. Scherer, and U. Brüning, *A Resource Optimized Remote-Memory-Access Architecture for Low-latency Communication*, in *2009 International Conference on Parallel Processing*, 2009, pp. 220–227
- [18] C. Leber, *Efficient Hardware For Low Latency Applications*, Dissertation, University of Mannheim, 2012
- [19] H. Fröning and H. Litz, *Efficient hardware support for the Partitioned Global Address Space*, in *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, 2010, pp. 1–6
- [20] M. Müller, *Exploring the Testability Methodology and the Development of Test and Debug Functions for a Complex Network ASIC*, Diploma Thesis, University of Mannheim, 2011
- [21] P. Kermani and L. Kleinrock, *Virtual cut-through: A new computer communication switching technique*, *Computer Networks*, vol. 3, no. 4, pp. 267–286, 1979
- [22] N. Burkhardt, *Fast Hardware Barrier Synchronisation for a Reliable Interconnection Network*, Diploma Thesis, University of Mannheim, 2007
- [23] M. Nuessle, H. Froening, S. Kapferer, and U. Bruening, *Proposal: Accelerate Communication, not Computation!*, in *High-Performance Computing Using FPGAs*, Springer London, Limited, 2013, p. 400

- [24] B. Chandrasekaran, D. Buntinas, S. Kini, D. K. Panda, and P. Wyckoff, *Microbenchmark performance comparison of high-speed cluster interconnects*, *IEEE Micro*, vol. 24, no. 1, pp. 42–51, Jan. 2004
- [25] Semico Research Corporation, *How an FPGA Approach to Complex System Design Can Improve Profitability: Real Case Studies*, 2012
- [26] Altera Corporation, *HardCopy IV Device Handbook*, vol. 1.
- [27] Taiwan Semiconductor Manufacturing Company Inc., *TSMC Announces Multi-layer Mask Service*, 2007
- [28] *ChipEstimate.com - Chip Planning Portal and IP Catalog*, <http://www.chipestimate.com/>, [last accessed: 01-Oct-2012]
- [29] IEEE Computer Society, *1364-2005 - IEEE Standard for Verilog Hardware Description Language*. 2006
- [30] K. Chapman, *Get Smart About Reset : Think Local, Not Global*, vol. 272. pp. 1–7, 2008
- [31] K. Chapman, *Get your Priorities Right – Make your Design Up to 50 % Smaller*, vol. 275. pp. 1–9, 2007
- [32] C. Cummings, *Synchronous Resets? Asynchronous Resets? I am so confused! How will I ever know which to use?*, *Synopsys Users Group Conference, San Jose*, 2002
- [33] C. Cummings and D. Mills, *Asynchronous & synchronous reset design techniques-part deux*, *SNUG Boston 2003*, 2003
- [34] Xilinx Inc., *7 Series FPGAs Memory Resources User Guide*, 2012
- [35] A. J. Van De Goor, *Using march tests to test SRAMs*, *IEEE Design & Test of Computers*, vol. 10, no. 1, pp. 8–14, Mar. 1993
- [36] S. E. Schuster, *Multiple word/bit line redundancy for semiconductor memories*, *IEEE Journal of Solid-State Circuits*, vol. 13, no. 5, pp. 698–703, Oct. 1978
- [37] M. Spica and T. M. Mak, *Do we need anything more than single bit error correction (ECC)?*, *Records of the 2004 International Workshop on Memory Technology, Design and Testing*, 2004., pp. 111–116, 2004

## Bibliography

---

- [38] R. Naseer and J. Draper, *Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs*, in *ESSCIRC 2008 - 34th European Solid-State Circuits Conference*, 2008, pp. 222–225
- [39] R. C. Bose and D. K. Ray-Chaudhuri, *On a class of error correcting binary group codes*, *Information and Control*, vol. 3, no. 1, pp. 68–79, Mar. 1960
- [40] A. Hocquenghem, *Codes correcteurs d'erreurs*, *Chiffres (Paris)*, vol. 2, pp. 147–156, 1959
- [41] I. S. Reed and G. Solomon, *Polynomial Codes Over Certain Finite Fields*, *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, Jun. 1960
- [42] R. W. Hamming, *Error detecting and error correcting codes*, *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950
- [43] M. Y. Hsiao, *A Class of Optimal Minimum Odd-weight-column SEC-DED Codes*, *IBM Journal of Research and Development*, vol. 14, no. 4, pp. 395–401, Jul. 1970
- [44] *The Perl Programming Language*, <http://www.perl.org/>, [last accessed: 09-Oct-2012]
- [45] Motorola, *M68HC11 Reference Manual*. 2007
- [46] NXP Semiconductors, *I2C-bus specification and user manual*.
- [47] Xilinx Inc., *Chipscope Pro and Cores*, 2012
- [48] Altera Corporation, *Design Debugging Using the SignalTap II Logic Analyzer*, in *Quartus II Handbook v12.0*, 2012, pp. 1517–1588
- [49] M. Arora, *Clock Dividers Made Easy*, *SNUG Boston 2002*, pp. 1–19
- [50] F. H. J. Feldbrugge, *Johnson counter circuit with invalid counter position detection and correction mechanism*, U.S. Patent US49930511989
- [51] Y. Zhang, *Design and Implementation of a PLL Clock Generator in 65nm CMOS*, Master Thesis, RWTH Aachen, 2010
- [52] Accelera, *Universal Verification Methodology (UVM) 1.1 User's Guide*. 2011
- [53] IEEE Computer Society, *1800-2009 - IEEE Standard for System Verilog-Unified Hardware Design, Specification, and Verification Language*. 2009

- [54] N. Burkhardt, *A Hardware Verification Methodology for an Interconnection Network with Fast Process Synchronization*, Dissertation, University of Mannheim, 2012
- [55] Intel Corporation, *Intel MPI Benchmark*, <http://software.intel.com/en-us/articles/intel-mpi-benchmarks>, [last accessed: 30-Oct-2012]
- [56] H. Fröning, M. Nüssle, D. Slognat, H. Litz, and U. Brüning, *The HTX-Board: A Rapid Prototyping Station*, 3rd annual FPGAworld Conference, 2006
- [57] P. R. Schulz, U. Brüning, and G. Strube, *SEED2002 support of educational course for electronic design*, in *Proceedings 2003 IEEE International Conference on Microelectronic Systems Education. MSE'03*, pp. 53–54
- [58] Cadence Design Systems, *Encounter RTL Compiler*, [http://www.cadence.com/products/ld/rtl\\_compiler/pages/default.aspx](http://www.cadence.com/products/ld/rtl_compiler/pages/default.aspx), [last accessed: 21-Nov-2012]
- [59] Cadence Design Systems, *Encounter Digital Implementation System*, [http://www.cadence.com/products/di/edi\\_system/pages/default.aspx](http://www.cadence.com/products/di/edi_system/pages/default.aspx), [last accessed: 21-Nov-2012]
- [60] I. Synopsys, *Liberty User Guides and Reference Manual Suite*. 2011
- [61] Cadence Design Systems, *LEF / DEF Language Reference*, 2012
- [62] Synopsys Inc., *Using the Synopsys® Design Constraints Format*, 2010
- [63] *Apache Subversion*, <http://subversion.apache.org/>, [last accessed: 01-Aug-2012]
- [64] J. Cong, Z. Pan, L. He, C.-K. Koh, and K.-Y. Khoo, *Interconnect design for deep submicron ICs*, in *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD) ICCAD-97*, 1997, pp. 478–485
- [65] Y. Zorian and A. Yessayan, *IEEE 1500 utilization in SOC design and test*, in *IEEE International Conference on Test, 2005.*, pp. 543–552
- [66] D. Sylvester and K. Keutzer, *A global wiring paradigm for deep submicron design*, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 242–252, 2000

## Bibliography

---

- [67] S. Scott, *Challenges and opportunities in the post single-thread-processor era*, *Proceedings of the 15th international conference on Parallel architectures and compilation techniques - PACT '06*, pp. 63–63, 2006
- [68] K. Shahookar and P. Mazumder, *VLSI cell placement techniques*, *ACM Computing Surveys*, vol. 23, no. 2, pp. 143–220, Jun. 1991
- [69] E. Wein and J. Benkoski, *Hard macros will revolutionize SoC design*, *EETimes*, 2004
- [70] N. A. Sherwani, *Algorithms for VLSI physical design automation*. Kluwer Academic Publishers, 1999, p. 572, ISBN 0792383931
- [71] A. B. Kahng, *Classical floorplanning harmful?*, in *Proceedings of the 2000 international symposium on Physical design - ISPD '00*, 2000, pp. 207–213
- [72] A. Pullini, F. Angiolini, S. Murali, D. Atienza, G. De Micheli, and L. Benini, *Bringing NoCs to 65 nm*, *IEEE Micro*, vol. 27, no. 5, pp. 75–85, Sep. 2007
- [73] E. Bogatin, *Signal and Power Integrity - Simplified (2nd Edition)*. Prentice Hall, 2009, p. 792, ISBN 0132349795
- [74] *IBTA - InfiniBand Trade Association*, <http://www.infinibandta.org>, [last accessed: 06-Nov-2012]
- [75] PCI-SIG, *PCI Express Card Electromechanical Specification*, Revision 2. 2007
- [76] Samtec Inc., *High Density High Speed I/O System HDI6, HDC Series*.
- [77] HyperTransport Technology Consortium, *HyperTransport<sup>TM</sup> Node Connector Specification*. 2009
- [78] IPC, *IPC-2221 Generic Standard on Printed Board Design*. 1998, pp. 1–123
- [79] J. A. Mears, *National Semiconductor Application Note 905*. 1996
- [80] E. Bogatin, *Roadmaps of Packaging Technology*. Integrated Circuit Engineering, 1997, ISBN 1-877750-61-1
- [81] JEDEC JC-11 Committee, *JEDEC Registered and Standard Outlines for Solid State and Related Products (JEP95)*.

- [82] L. Wang, Z. Zhao, Q. Wang, and J. Lee, *Characterize the microstructure and reliability of ultra fine pitch BGA joints*, in *2009 International Conference on Electronic Packaging Technology & High Density Packaging*, 2009, pp. 674–678
- [83] KYOCERA SLC Technologies, <http://www.kyocera-slc.co.jp/>, [last accessed: 13-Nov-2012]
- [84] KYOCERA SLC Technologies, *Organic Substrate HDBU Design Guideline*. 2010
- [85] J. H. Lau, *Flip chip technologies*. McGraw-Hill, 1996, p. 565, ISBN 0070366098
- [86] R. R. Tummala, E. J. Rymaszewski, and A. G. Klopfenstein, *Microelectronics Packaging Handbook, Part 2: Semiconductor Packaging (Pt. 1)*. Springer, 1997, p. 1030, ISBN 0412084414
- [87] Taiwan Semiconductor Manufacturing Company Inc., *TSMC 65NM/55NM CMOS LOGIC/MS\_RF DESIGN RULE*. 2009, ISBN 1202006140
- [88] Cadence Design Systems, *Allegro Package Designer*, [http://www.cadence.com/products/pkg/package\\_designer/pages/default.aspx](http://www.cadence.com/products/pkg/package_designer/pages/default.aspx), [last accessed: 21-Nov-2012]
- [89] Agilent, *Advanced Design System (ADS)*, <http://www.home.agilent.com/en/pc-1297113/advanced-design-system-ads>, [last accessed: 21-Nov-2012]
- [90] M. Magin, *Power and Signal Analysis of a High Performance ASIC*, Master Thesis, University of Heidelberg, 2013
- [91] Altera Corporation, *Minimizing Ground Bounce & VCC Sag Whitepaper*
- [92] IEEE Computer Society, *1149.1-1990 - IEEE Standard Test Access Port and Boundary - Scan Architecture*. 1990
- [93] ITRS, *Test and Test Equipment*, in *International Technology Roadmap for Semiconductors*, 2011
- [94] S. Kundu, T. M. Mak, and R. Galivanche, *Trends in manufacturing test methods and their implications*, in *2004 International Conference on Test*, pp. 679–687
- [95] S. Kim, D. Kong, C. Cho, J. Nam, B. Kim, and J. Lee, *Design and fabrication of MEMS test socket for BGA IC packages*, in *2010 IEEE Sensors*, 2010, pp. 1896–1899

## Bibliography

---

- [96] C. G. Bell, J. C. Mudge, and J. E. McNamara, *Computer Engineering: A DEC View of Hardware Systems Design*. Digital Press, 1978, p. 585, ISBN 0932376002
- [97] V.-K. Kim, T. Chen, and M. Tegethoff, *ASIC manufacturing test cost prediction at early design stage*, *Proceedings International Test Conference 1997*, pp. 356–361, 1997
- [98] D. K. deVries, *Investigation of Gross Die Per Wafer Formulas*, *IEEE Transactions on Semiconductor Manufacturing*, vol. 18, no. 1, pp. 136–139, Feb. 2005
- [99] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability (The Morgan Kaufmann Series in Systems on Silicon)*. Morgan Kaufmann, 2006, p. 808, ISBN 0123705975
- [100] IEEE Computer Society, *IEEE Std 1500™-2005 IEEE Standard Testability Method for Embedded Core-based Integrated Circuits*, no. August. 2005, ISBN 0738146935
- [101] V. Iyengar, T. Yokota, K. Yamada, T. Anemikos, B. Bassett, M. Degregorio, R. Farmer, G. Grise, M. Johnson, D. Milton, M. Taylor, and F. Woytowich, *At-Speed Structural Test For High-Performance ASICs*, in *2006 IEEE International Test Conference*, 2006, pp. 1–10
- [102] P. Girard, N. Nicolici, and X. Wen, *Power-Aware Testing and Test Strategies for Low Power Devices*, vol. 2009. Springer, 2009, p. 353, ISBN 1441909273
- [103] ITRS, *Design*, in *International Technology Roadmap for Semiconductors*, 2011
- [104] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits (2nd Edition)*. Prentice Hall, 2003, p. 761, ISBN 0130909963
- [105] N. Badereddine, P. Girard, S. Pravossoudovitch, C. Landrault, A. Virazel, and H.-J. Wunderlich, *Minimizing peak power consumption during scan testing: test pattern modification with X filling heuristics*, in *International Conference on Design and Test of Integrated Systems in Nanoscale Technology, 2006. DTIS 2006.*, 2006, pp. 359–364
- [106] IEEE Computer Society, *1450-1999 - IEEE Standard Test Interface Language (STIL) for Digital Test Vector Data*. 1999
- [107] ST Microelectronics, *STM32F405xx / STM32F407xx Datasheet*. 2012
- [108] Future Technology Devices International Ltd., *FT2232H - Hi-Speed Dual USB UART/FIFO IC Datasheet*.

- [109] P. Rashinkar, P. Paterson, and L. Singh, *System-on-a-Chip Verification: Methodology and Techniques*. Springer, 2000, p. 392, ISBN 0792372794
- [110] Wilson Research Group and Mentor Graphics, *2010 Functional Verification Study*, 2010
- [111] P. Yeung and S. Choi, *Advanced static verification for SoC designs*, in *2009 International SoC Design Conference (ISOCC)*, 2009, pp. 295–300
- [112] *The Trac Project*, <http://trac.edgewall.org/>, [last accessed: 17-Dec-2012]



## C EXTOLL Package

Symbol	Millimeters	Description
A	tbd. <sup>10</sup>	Overall Height
A1	tbd.	Vertical Stand Off
A2	tbd.	Package Body Thickness
A3	tbd.	Height of Thermal Lid
b	0.60 ± 0.10	Ball Diameter
D	42.50	Package Body Length
D1	42.00	Distance between the centerlines of the two outermost columns of balls
E	42.50	Package Body Width
e	1.00	Distance between the centerlines of two adjacent balls
E1	42.00	Distance between the centerlines of the two outermost rows of balls

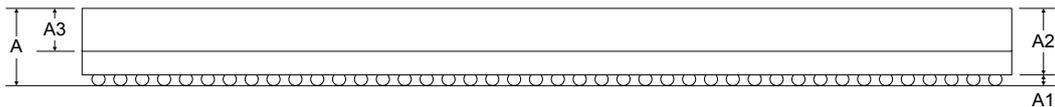


Figure C.1 Side View

<sup>10</sup> Substrate thickness is 1160µm

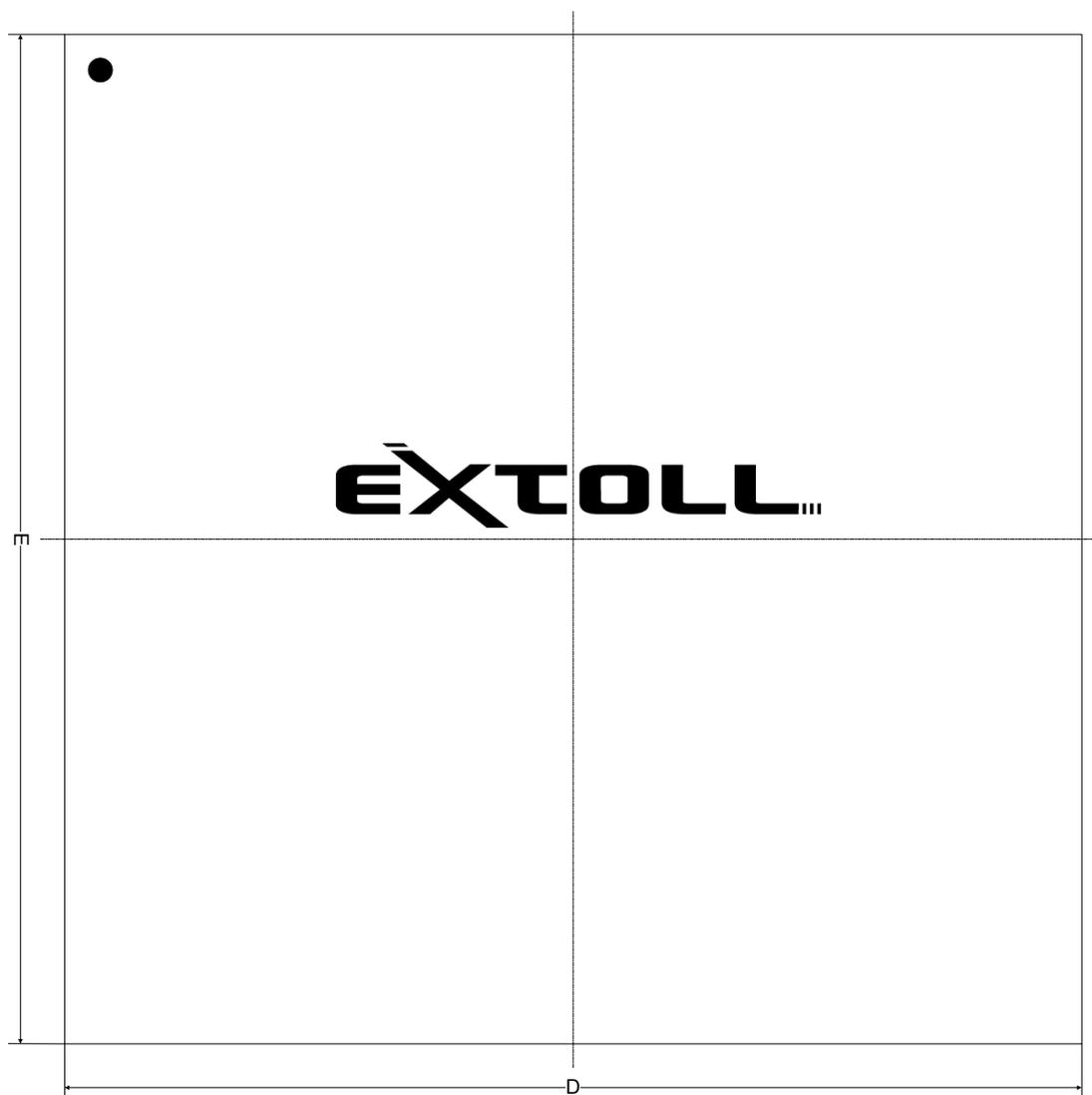


Figure C.2 Top View

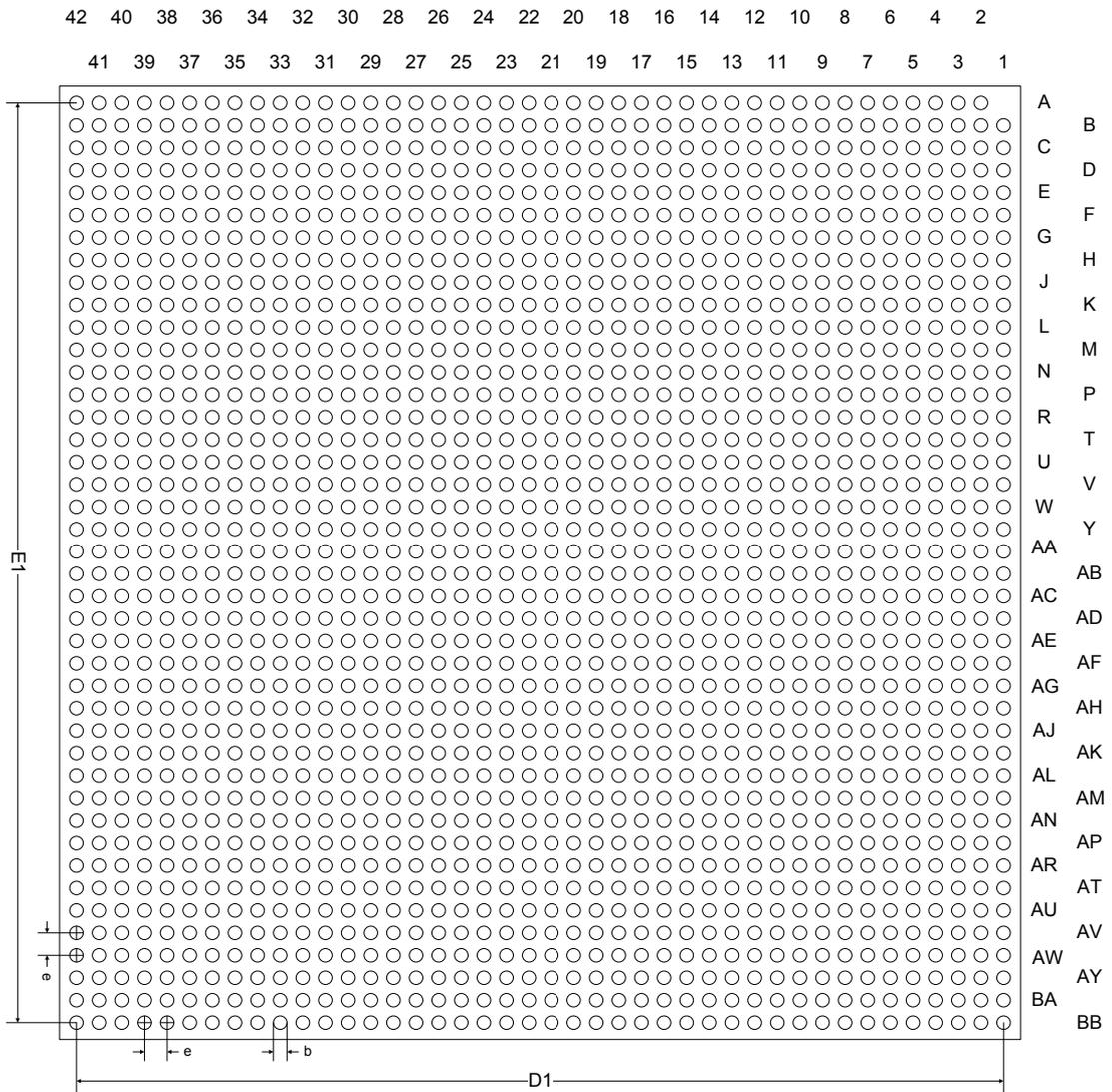


Figure C.3 Bottom View



	Link TX/RX		Digital Core GND
	HT TX/RX		Digital Core VCC
	PCIe TX/RX		Analog Serializer GND
	Not Connected		Analog Serializer VCC1
	Test Related Signals		Analog Serializer VCC2
	Link Detect		I/O VCC
	Global Config Signals		Analog GND
	GPIO		Analog VCC
	Debug Signals		PLL GND
	Oscillator		PLL VCC
	Backup Clock		HT Bias Supply
	Sideband Signals		HT Digital Core GND
	Resistor Calibration		HT Digital Core VCC
			HT Digital GND
			HT Digital VCC
			HT Analog GND
			HT Analog VCC



## D STIL Example

```

STIL 1.0;

Signals {
  A0 InOut;
  B0 InOut;
  TC In;
}

SignalGroups {
  ALL = 'A0 + B0' + TC;
  SI1 = 'A0' { ScanIn 10; }
  SO1 = 'B0' { ScanOut 10; }
  TCK = 'TC';
}

Timing {
  WaveformTable scan_wf {
    Period '100ns';
    Waveforms {
      ALL { 10 { '0ns' U/D; }}
      ALL { HLZX{ '0ns' Z; '50ns' H/L/T/X; }}
      TCK { P { '0ns' D; '10ns' U; '60ns' D; }}
    }
  }
}

PatternBurst "scan_burst" {
  PatList { "scan"; }
}

PatternExec {
  PatternBurst "scan_burst";
}

MacroDefs {
  "scan" {
    W scan_wf;
    C { TCK=P; SI1=0; SO1=X; }
    Shift { V { SI1=#; SO1=#; } }
    C { TCK=0; }
  }
}

Procedures {
  "scan" {
    W scan_wf;
    V { ALL=0Z0; }
    Shift { V { TCK=P; SI1=#;SO1=#; }}
  }
}

Pattern "scan" {

```

## STIL Example

---

```
W scan_wf;
V { ALL=0Z0; }
Macro "scan" {
  SI1=0000000000; }
V { TCK=P; }
Call "scan" {
  SO1=LLLLLLLLLL; }
}
```

The STIL file is divided in several sections.

The first section “Signals” defines the available pins and their directions. In this example two bidirectional pins are defined, A0 and B0 as well as an input signal TC.

In the next section “SignalGroups” the signals defined before are added to logical groups so that they can be referenced later on. This example defines a signal vector ALL that concatenates all signals as well as SI and SO with a single member each (A0 / B0) that are defined as scan input and scan output with a scan chain depth of 10. At last TCK is defined with the single member TC.

In the “Timing” sections so called waveforms can be defined that describe how signals are processed. In the example, the basic clock cycle is set to 10 MHz (100ns). For the ALL vector two behaviors are assigned. In the case of a 1 or a 0 in the datastream the waveform goes up/down immediately. If a H,L,Z or X is used there is a 50ns wait so that the data value can be sampled in the middle of the clock cycle. The value will be mapped to H(igh), L(ow), T(ristate) or X(Unknown). For TCK a pulse (P) behavior is defined to generate a clocking event.

Because of the simplicity of the example the next two sections are rather short. The “PatternBurst” section only calls one scan pattern that is defined later in the file. The “PatternExec” section in turn also only calls the one “PatternBurst” that was defined.

The next two sections are similar. One defines a macro that is used to shift scan patterns in, the other is a procedure that captures data at the scan outputs. Both are then used in the “Pattern” section. Once again, the example only defines a single construct. The pattern calls the macro “scan” to shift ten consecutive zeros into the scan chain. Afterwards the clock is pulsed for one cycle and the procedure “scan” is called to shift the patterns out. The expected return value for this operation is the same as the input, ten consecutive zeros.

However, the example above is only a short excerpt of the capabilities of the STIL language to give some insight to the syntax. More information can be found in the official specification.