# A Reconfigurable Accelerator Card for High Performance Computing

**Michael James Aitken**
Supervisor: Professor Michael Inggs
Co-supervisor: Dr Alan Langman

A dissertation submitted to the Department of Electrical Engineering in fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

at the

UNIVERSITY OF CAPE TOWN

November 2008

# Declaration

I know the meaning of plagiarism and declare that all the work in this thesis, save for that which is properly acknowledged, is my own. It is being submitted for the degree of Master of Science in Electrical Engineering at the University of Cape Town. It has not been submitted before for any degree or examination in any other university.



Signature of Author .....

Cape Town
25th November 2008

# Abstract

This thesis describes the design, implementation, and testing of a reconfigurable accelerator card. The goal of the project was to provide a hardware platform for future students to carry out research into reconfigurable computing.

Our accelerator design is an expansion card for a traditional Von Neumann host machine, and contains two field-programmable gate arrays. By inserting the card into a host machine, intrinsically parallel processing tasks can be exported to the FPGAs. This is similar to the way in which video game rendering tasks can be exported to the GPU on a graphics accelerator. We show how an FPGA is a suitable processing element, in terms of performance per watt, for many computing tasks.

We set out to design and build a reconfigurable card that harnessed the latest FPGAs and fastest available I/O interfaces. The resultant design is one which can run within a host machine, in an array of host machines, or as a stand-alone processing node. The final design had the following specifications:

| | |
|---|---|
| **Form factor** | HyperTransport Expansion Card Standard Rev1.1 |
| **On-board FPGAs** | Xilinx Virtex-5 LX110T-FF1136 and LX50-FF676 |
| **I/O Capabilities** | 16-lane HTX connector, 2 × 10GBASE-CX4 connectors |
| **On-board memory** | 6 × Samsung QDRII+ |
| **Power Consumption** | minimum 7W, maximum 50W |

We discuss these design choices in light of our reviews of similar commercial systems. We then explain how Mentor Graphics tools were used to produce a complete design specification which was packaged and sent for layout design at Mechatronics Test Equipment. We simulated the resultant layout in software and found it to be problem free. Fabrication of the PCBs took place in California, whilst component assembly was carried out by a local firm. Various gateware test benches were produced for checking the validity of the on-board interfaces such as the 10GBASE-CX4s and the QDRII+ modules, and we carried out probe tests with an oscilloscope. The assembled boards were found to be entirely functional according to their design specification.

# Acknowledgements

I would like to acknowledge a number of people who helped me considerably during the course of my project.

Firstly, to my supervisors:
I thank Professor Mike Inggs for his endless efforts in organising funding, providing resources, and dishing out wisdom and advice to make the project possible. I am grateful for the way this project was made significant to the new ACE research group. Also, Dr Alan Langman donated much of his time to help flesh out the design issues and was also kind enough to put up with my presence at the SKA/KAT office for many months.

To my fellow masters students at the SKA and the CHPC:
Andrew Woods, Peter McMahon, Jason Manley, Jason Salkinder, Nick Thorne, Jane Hewitson, Michael Gorven and Joe Milburn - thanks for all the laughs and good times over the last two years.

To the engineers and administration staff of the SKA/KAT:
Francois Kapp, David George and Alec Rust - thank you for hours of help and advice on PCBs as well as FPGA related issues. Kim de Boer and Annah Mashemola - you made the experience of being a SKA/KAT student a very pleasant one.

To Saira Shaikh of MTE:
Thank you for your very professional work and prompt email responses that helped create a successful layout design.

To Xilinx, Inc.
Thank you for the donation of the Virtex-5 FPGAs, without which this project would have been enormously expensive.

To the employees at Tellumat:
Ashwin Kasi and others - thank you for your great work in assembling the PCBs and for sorting through the most complicated mess of components you've probably ever seen.

My two years of living, travel and equipment expenses were funded by my SKA postgraduate bursary, and I am truly grateful for the resources and experiences I was afforded through the SKA - Human Capital Development Programme.

To anyone else I have forgotten to mention here, thank you!

# Contents

# List of Figures

# List of Tables

# Nomenclature

| | |
|---|---|
| API | Application Programming Interface |
| ASIC | Application-Specific Integrated Circuit |
| ATX | Advanced Technology Extended - a form factor standard for computer case and motherboard designs |
| BGA | Ball Grid Array - a type of surface-mount packaging used for integrated circuits |
| Bit-stream | the configuration data for an FPGA |
| BTX | Balanced Technology Extended - a newer form factor standard for computer cases and motherboards, and a replacement for ATX |
| CMOS | Complementary Metal-Oxide-Semiconductor, is a common class of intergrated circuits. |
| CPU | Central Processing Unit |
| DDR | Double Data Rate |
| FPGA | Field-Programmable Gate Array |
| Gateware | HDL source code that is specific to a particular FPGA architecture |
| Gerber File | a standard file format used to describe a layer of a printed circuit board design |
| GPU | Graphics Processing Unit |
| HSTL | High-Speed Transceiver Logic |
| HT | HyperTransport |
| HTX | HyperTransport Expansion Standard |
| IBIS | Input Output Buffer Information Specification |
| IC | Integrated Circuit |

| | |
|---|---|
| IEEE | Institute of Electrical and Electronics Engineers |
| JTAG | Joint Test Action Group - a standard for accessing ICs on a printed circuit board for testing purposes. |
| LVCMOS | Low Voltage Complementary Metal Oxide Semiconductor |
| LVDS | Low Voltage Differential Signalling |
| MAC | Media Access Control |
| MGT | Multi-Gigabit Transceiver |
| MTE | Mechatronics Test Equipment - a company in Pune, India who provide PCB design services |
| PCB | Printed Circuit Board |
| PCIe | PCI Express or Peripheral Component Interconnect Express |
| PHY | a common abbreviation for the physical layer of a networking standard |
| PROM | Programmable Read-Only Memory |
| PSU | Power Supply Unit |
| RAM | Random Access Memory |
| RoHS | Restriction of Hazardous Substances Directive |
| SFP | Small Form-Factor Pluggable connector |

# Chapter 1

# Introduction

This thesis describes the investigation into, the design, the implementation, and the testing of a reconfigurable accelerator card to be used for high performance computing. The content of this thesis is almost entirely practical, and thus it reads like a journal, in that it describes in a somewhat chronological order, the work that was carried out during the course of the project.

In this chapter we describe our objectives and motivation, after which we summarise each of the remaining chapters. A background to the project objectives is presented in detail in Chapter 2.



## 1.1 Objectives

In this project, the main objectives were to:

- build a reconfigurable accelerator card[1] to be used for high performance computing
- provide a basic set of gateware to test the functionality of the design
- test the card's interfaces using the developed gateware
- make recommendations for future work to develop the card into a more complete computing platform

In order to accomplish these primary objective they were broken down into the following sub-objectives, listed from 1 to 6:

1. **Identify the key requirements of a reconfigurable accelerator**
   In order to define the specifications of a new accelerator card, we required a thorough investigation of the similar devices currently in production and what

[1] A reconfigurable accelerator card is an adaptor card for a traditional host computing system. The card is populated with one or more reconfigurable devices, such as an FPGA, which can be used by the host to offload various computational tasks. A detailed explanation is given in Section 2.4

type of similar functionality we wanted to implement. This entailed borrowing ideas and concepts from previous work and incorporating those ideas into a new improved device. We aimed to look at the various configurations in which a reconfigurable accelerator card would be used and then defined the requirements to fulfil those needs.

2. **Investigate the applicable technologies for reconfigurable co-processing**
   Reconfigurable co-processors make use of a multitude of different technologies and standards, which all have a material effect on their performance. There are various FPGA devices to choose from, and different I/O standards which offer different advantages and disadvantages. A thorough investigation of the various options was required in order to satisfy the requirements specified in step 1.

3. **Create a conceptual design for a reconfigurable accelerator card**
   The first step in the design capture was to draw up a conceptual design which could be scrutinised for any functional flaws, and which could be checked against the requirements specified in step 1. The conceptual design needed to specify the components and logical connections, taking into account pin counts, I/O standards and device capabilities. The design needed to harness the technologies identified during step 2.

4. **Implement the hardware design**
   After refining the conceptual design, the precise low-level schematic design needed to be captured using a CAD tool. This process defined the exact part numbers and pin allocations for the physical layout, whilst taking into consideration the requirements of the high speed data transmission signals. Additionally the schematic served as a pin map for development of the gateware.

5. **Implement basic gateware for testing**
   In order to test the functionality of an assembled card, a basic set of HDL controller cores was needed for compilation into bit-streams for the FPGAs. The aim was to verify the functionality of the various interfaces in the design.

6. **Specify test benches and verify the hardware**
   Using the gateware created in step 5, we created gateware test bench setups and tested the functionality of the interfaces on an individual basis, recording the results.

## 1.2 Motivation

The South African government wished to invest in local technological infrastructure as well as develop human resources across the engineering fields. In particular, the government had demonstrated its desire to create a niche environment for high performance computing in South Africa, most notably through its investment in the Karoo Array Telescope project[2], and the Centre for High Performance Computing[3], where the most powerful supercomputer in Africa had recently been installed. This project fell neatly into the goals of creating computing infrastructure, and building knowledge in this specialist field. The knowledge obtained, generated, shared, and documented during the course of this project will be of benefit to the South African computing initiative, and its success will encourage further work in reconfigurable computing, and hopefully help inspire tomorrow's students to become more involved with technology.

Current commercial reconfigurable accelerators are expensive devices which come bundled with proprietary software suites[4]. It made sense for the local academic community to have access to an open-source accelerator, whilst at the same time harnessing the latest improvements in FPGAs and I/O standards in a superior computing device. By providing a reconfigurable accelerator at fabrication price, future students may take the device and tool set, possibly improve them, and make use of an array of cards to build arbitrary reconfigurable computing clusters. Indeed the concept of an open-source accelerator had already been implemented[1, 2]. We simply wished to produce the next generation of such a device, borrowing from and improving on those same ideas.

## 1.3 Chapter Summaries

The remaining chapters of this dissertation are briefly summarised in the paragraphs below:

**Chapter 2** presents the technical background to this project. We discuss the reemergence of parallel computing due to the technology walls that are being encountered in strenuous computing tasks. We give a simple formula for power consumption which best summarises the issue facing high clock frequencies in electronic circuits. Then in terms of parallel processing, we mention the effect known as Amdahl's Law, and show how it influences application performance as the number of processing elements are increased. We then introduce reconfigurable computing, and mention how it may in some cases bypass this effect. Finally we move on to describe the important concepts of a reconfigurable co-processor.

[2] `http://www.ska.ac.za`
[3] `http://www.chpc.ac.za`
[4] This refers to products from *Nallatech, Celoxica, AlphaData*, amongst others.

**Chapter** 3 discusses the latest technologies which are applicable to building a reconfigurable co-processor. We provide a technology review of the existing reconfigurable accelerator cards, specifically Celoxica's RCHTX card, Nallatech's BenONE, Pico Computing's PICO E-16 and the University of Mannheim's HTX-Board. We consider the different approaches taken by these cards and the key architectural choices made by their designers: the type of FPGA devices, I/O interfaces, memories types and the bundled software resources. A breakdown of the important differences and similarities between these different devices is provided.

Next we discuss the latest FPGA technology giving special attention to the market leaders *Xilinx* and *Altera*, whilst also touching on the products of other manufacturers such as *Actel* and *Lattice*. The chapter focuses specifically on Virtex-5 and Stratix-III families of FPGAs, as these are the latest products available. We consider the various hardwired resources that these top-of-the-range devices offer, as well as their suitability to the project goal.

We take a look at the all important choice of the interface with which to couple the accelerator card to a host machine. A comparison is made between two leading expansion interfaces: HyperTransport and PCI-Express. We present the theoretical performances and the limitations of these two standards, and discuss their suitability to co-processing, concluding that HyperTransport offers the best performance in terms of latency and throughput. We also mention the variation in levels of support available for the two standards.

We go on to discuss the types of on-board memory that the existing products make use of, and compare the strengths and weaknesses of the various options, such as QDR SRAM, ZBT SRAM amd DDR DRAM. We mention which categories of performance the various technologies are placed in, and discuss their different applications.

We go on to discuss the possible back-end I/O capabilities of an accelerator card, and how they will effect the cards usability. Specifically we talk about the standards 10GbE, InfiniBand and we weigh up the benefits of copper vs fibre interconnects.

Where the information is available, we analyse the power consumption of the different devices, in light of their I/O, computational ability and possible cooling requirements.

**Chapter** 4 discusses the requirements of a general reconfigurable co-processor in light of the technologies available, and the discussions in Chapter 3. First we decide on the form factor of the card. We explain why an expansion card for an ATX or BTX host machine is the most desirable form factor from a mechanical, design time and usability point of view.

We then go on to define the functional requirements by first discussing possible use cases in various applications, and the architecture that will suit such general processing requirements. A decision is made that the card will have multiple high-throughput back-end I/O interfaces, due to the desire for card-to-card and card-to-

switch interfacing. We present scenarios where such a feature would be useful and we illustrate these ideas.

The choice of reconfigurable logic is discussed, and we conclude that the latest and most resourceful FPGAs are without doubt the ideal type of reconfigurable logic for the accelerator card.

The latency between the co-processor and its host node is identified as an important issue, and we decide that the expansion standard with the lowest latency would make a compelling choice as the standard on which to base the accelerator card's main interface. We argue against using high capacity high latency RAM on the accelerator card itself due to the card's direct coupling to system RAM, and instead suggest that a low-latency, high-throughput RAM is chosen for on-board memory.

Furthermore, we discuss the issue of power consumption, mostly focused on dealing with the power dissipated by the on-board FPGAs, and the limitations this places on the possible designs for the card. Lastly, we briefly consider the cost of the accelerator card, mentioning why cost is an unreliable metric in such a cutting-edge field.

**Chapter 5** presents the decision making that went into producing the conceptual hardware design, with a break down on reasons for including the various interfaces and silicon devices. The inclusion of each device and interface is supported by an explanation of it's capabilities in light of the requirements specified in Chapter 4.

The convenience of choosing Xilinx FPGAs is discussed in light of their availability in South Africa and the Xilinx University Donation Program. We move on the describe the exact devices (Xilinx Virtex-5 LX50 and Virtex-5 LX110T) that were to be used on the card, and the paired configuration that they would be implemented in. We point out how this concept was borrowed from Celoxica's RCHTX-XV4 board, and how it allows the LX110T FPGA to be reconfigured via the LX50 FPGA during run-time. We show how the LX50 FPGA device can be configured at power-up using a Xilinx PROM device. The JTAG debugging and programming chain is presented, along with a diagram and explanations as to its path around the board.

Next, we demonstrate how a HTX HyperTransport port will be used to interface the card to its host node. The card will consist of a 16-lane HyperTransport link directly connecting the on-board FPGAs to the host machine. This HTX standard connector also delivers a 12V supply directly to the card from the host motherboard. This optional source of power is discussed more accurately in section 5.2.

Next the choice of on-board QDRII+ memory devices is explained with reference to the requirements. We show how the QDRII+ devices are connected to the two FPGAs, and why we chose to use six memory devices with 18-bit interfaces. We move on to present the 10GBASE-CX4 interfaces that couple the large LX110T FPGA to the board's back-end, illustrating how these ports can in future be used to construct a reconfigurable cluster, as well as allow the streaming of large data

blocks. The choice of copper CX4 connectors versus more expensive fibre offerings is explained in light of various factors such as space, cost and availability.

We explain that the board will need a serial UART interface for simple debugging, and show how it will be interfaced with the LX50 FPGA.

The various power regulators are then discussed, and we explain how the board can be powered via either the HTX interface or an auxiliary PCIe 6-pin connector. The spreadsheets used for calculating the total various power requirements, decoupling, and other design criteria are also presented.

The overall design is then shown in a complete conceptual plan, along with a diagram that shows the power regulators and the power network.

**Chapter 6** discusses the process of capturing the detailed system schematic, sparing the reader from the tedious elements of CAD design. The *Mentor Graphics* CAD suite is presented as the software package to be used to capture the design. Next we describe the FPGA pin allocation procedure using *Mentor Graphics I/O Designer*, where the large number of FPGA pins were handled. We focus on how the pins where allocated to different FPGA I/O banks, taking into consideration the convenience of laying out the board, as well as the allocation of clock signals to clock capable pins within each bank.

We then move on to describing the use of DxDesigner to capture the pin mappings into symbols for each of the various chosen devices. We describe how the entire system design was partitioned into subsystems based on function and how these sub-systems appeared in the schematic hierarchy.

An important part of the schematic capture process was calculating the required decoupling capacitance and inductance required for adequate voltage integrity throughout the board. This was done by closely following a *Texas Instruments* white-paper on regulator decoupling, as well as the official *Xilinx* FPGA decoupling recommendations.

The process of encapsulating the entire design specification in preparation for outsourcing the layout is out-lined. We discuss the decision for outsourcing the design review of the revision 1 schematic as well as the layout component of the design work to the *Mechatronics Test Equipment Pvt. Ltd.* in Pune, India. This was done in order to focus on other aspects of the design, such as gateware design. MTE was chosen because of their expertise with Virtex-5 FPGAs as well as a recommendation by colleagues at the University of Berkeley, California. As part of their service MTE conducted a design review of our Revision 1 schematic. We show how the bill of materials was drawn up and describe the process of sourcing the components. After the initial layout by MTE, we conducted signal integrity tests on the generated Gerber files using *Mentor Graphics Hyperlynx*.

We chose to have the PCBs manufactured at *StreamLineCircuits* and we discuss the cost of this process as well as the fabrication standard used in producing the prototype boards. The number of printed boards was five, whilst the total number

of boards required to be assembled was two, the difference being to allow for one PCB for a pre-assembly destructive test, and additional PCBs to assemble at a later stage.

The process of assembling the components onto the board was carried out by *Tellumat*, based in Cape Town. The full assembly was done after the power regulators had been soldered onto and tested with one of the PCBs, and the various voltage points had been measured to insure voltage compliance.

**Chapter** 7 runs through the process of preparing gateware controller cores for the testing of the accelerator card's different interfaces, and presents the findings from individually running these cores on the card.

Initially we needed to check the functionality of the FPGAs, by trying to configure them with simple gateware that toggles the on-board LEDs in a manor which could also test the operation of the crystal oscillators. Both FPGAs were configured using a JTAG programming cable. This test showed that the FPGAs themselves, the LEDs and the clocks were working as specified. The same test was then repeated, but instead the PROM device was successfully programmed with the bit-stream for the LX50, and the board was power cycled, to test the programming interface between the LX50 and the PROM device.

Next we consider the basic serial UART interface for use in debugging the board. We present the open-source core that was used to control this interface. We created gateware that implements the Serial UART core, and connected the physical serial pins on the accelerator card to a host serial port on another machine. An attempt to transmit serial data across the link in both directions was made and the interface was found to be functional.

We look at testing the HT-link between the two FPGAs. The link consists of 32 differential tracks or 64 single parallel tracks between the two devices. We created simple user gateware to transmit data across this interface in both directions without preparing an actual HyperTransport core. The data-rates achieved are discussed.

Next we move to the QDRII+ core, which is an extension of Xilinx's free Memory Interface Generator 2.0 core. We show how the original QDRII core was changed to produce a QDRII+ capable core. The FPGAs were programmed with gateware which implements QDRII+ cores and a test bench on each of the QDRII+ memory interfaces. We attempted to write data to the memory devices, and then read the data back again, comparing the result with the original data to determine any error rates. We incrementally increased the clock rate until such time as the link failed, and found that all of the QDRII+ banks bar one were running correctly at 150MHz.

Xilinx's XAUI proprietary core was used to test the CX4 back-end interfaces. We discuss the configuration options of the GTP_DUAL Virtex-5 primitives and how they effect the interface's performance. A simple loop-back test was carried out by implementing XAUI gateware on each of the interfaces (A & B), and then continuously transmitting data frames from A to B, as well as from B to A. By

reading off the status bits of the core, as well as reading off the error count on a simple comparator, we could determine that both interfaces were locked and synchronised to their incoming data streams.

The University of Mannheim's Computer Architecture Group has developed an open-source 16bit HyperTransport core that we can use to implement the HT interface. The core was targeted at the Virtex-4 range of FPGAs, and as such requires the manipulation of HDL code to re-target it towards Virtex-5 FPGAs. The development of the HT core, as well as the software drivers was bundled as a separate project for another student, Nicholas Thorne, who has taken on the task of implementing the HT core.

The test results are tabulated to give a summary of the functionality of the board, along with the various significant measurements taken during the tests.

**Chapter 8** describes the future work that is recommended for the development of this project. The chapter is broken down into two separate sections: crucial work that needs to be undertaken to transform the accelerator card into a usable device, and other extensions to the product that will make it usable in foreseen circumstances and applications.

Initially, we consider all the identifiable problems with Revision 1 accelerator card, and suggest that a Revision 2 card is released in order make the necessary corrections.

Next we discuss the further development of an API suite that will provide C-style routines for transferring blocks or streams of data to the accelerator card and back again. In addition to this, we will require a mechanism to reconfigure the larger LX110T FPGA by configuring it with a new bit-stream using a configuration controller implemented in the LX50 FPGA. As with most APIs, we will require a fair amount of documentation to make it user-friendly.

Some additional extensions to the project are hinted at towards the end of the chapter. We present different cluster configurations under which the card could operate with the appropriate 10GbE switches.

# Chapter 2

# A Background to Reconfigurable Computing

## 2.1 The Rebirth of Parallel Computing

Parallel computing hails from a time before the personal computer. It has always played an important role in high performance computing, since it is a logical approach to accelerating the computation of the majority of real world programs. This is because some concurrency is found in almost every problem no matter what the application. In the time before Moore's Law [3] parallel processing was the key approach to tackling these problems, in order to beat the performance offered by purely sequential machines. IBM's 704 was the first recorded implementation of a parallel computer[4], followed by a number of relatively successful machines: the IBM 7030 Stretch supercomputer remained the fastest computer from 1961 until 1964. The early machines were generally over budget and underutilised due to the challenges of writing efficient code for them. The outlook for parallel computing changed with the dawn of the integrated circuit [5], and the continual decrease in transistor count and power consumption ever since. The economies of scale of personal computers and the reliable growth of general purpose CPU clock rates, drove down the cost of producing such devices. For more than twenty years frequency scaling was the computing paradigm, and the faith placed in the continual advancement of processor speeds meant that there was relatively little interest in producing intrinsically parallel machines.

However, behind the euphoria of the CPU silicon advancements lay the truth about the technology's physical limitations. The standard formula for dynamic power in a continuously switching CMOS gate is:

$$P = C \times V^2 \times F$$

where P is Power, C is the switching capacitance and F is the switching frequency of an individual gate [6]

The formula demonstrates that in order to increase the operating frequency of a given processor and keep the power consumption the same, one would have to decrease the capacitance in the same proportion by decreasing the transistor size. Thus, during the late 1990s, it became very obvious that the limits of copper/silicon technology would fairly soon be reached. Power consumption within a single silicon chip has become a major concern, along with the expenses of cooling and high device failure rates that go hand-in-hand with the heat dissipation. In addition to this, it's been found that manufacturing silicon devices with lower then 65nm fabrication technology can ultimately lead to internal soft error rates [7], introducing the complexity and expense of error correction techniques.

Thus, out of the conundrum of the frequency scaling paradigm, came the rebirth and return to parallel computing. In 2004 Intel announced that they would be halting their spending on the newer and faster (i.e. higher clock rate) replacements for the Pentium 4 and subsequently be investing in multi-core technology [8, 9]. This sealed the industry's fate in its retreat to the parallel paradigm. In turn, the renewed interest in parallel computing has boosted a host of smaller fields to provide new resources for high performance computing: multi-core processors, symmetric multiprocessing, clusters, massively parallel processing, grid computing, GPU processing and reconfigurable computing.

## 2.2 Encountering Amdahl's Law

Whilst working on many historical significant parallel computers Gene Amdahl began to notice a phenomenon that effected the utilisation of individual computing elements within a computing system. Many expected that as you increase the number of processing elements in a computer, the performance would scale linearly with the increase. This was not the case. In fact, as Amdahl demonstrated [10], most programs scaled linearly for a small number of elements at first, but then performance either remained constant or even diminished as more processing elements were added (see Figure 2.1).

This was attributed to the fact that most programs consist of both parallel and sequential parts, which are interdependent on each other for data. The law highlighted the fact that there was a theoretical limit to the performance gain that is possible for any given program due to these dependencies and the associated task management. Amdahl's law demonstrates the importance of optimising the data flow within a program to reduce the sequential operations to an absolute minimum, and importantly, it indirectly indicates the gains that can be had from using *intrinsically* parallel computing fabrics, rather then individual computing elements.

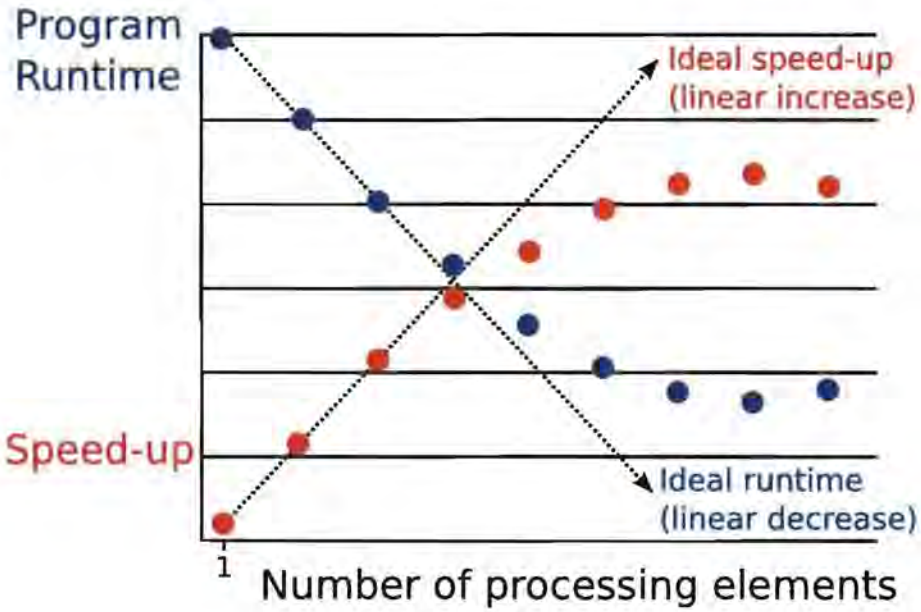



Figure 2.1: Demonstration of Amdahl's Law
*courtesy of Mark Pellegrini, University of Delaware*

## 2.3 The Unexploited Field of Reconfigurable Computing

Apart from the frequency scaling problem, normal Von Neumann computing systems also suffer from another noteworthy disadvantage, in that they are instruction-stream-based. This means that the CPUs are designed to increment through a predefined set of instructions, by incrementing program counters. Whilst this may seem to be a natural way of tackling any algorithm, in practice this makes it difficult to facilitate moving data between processors in a multi-processor environment, as we are measuring progress by using a program counter, rather then measuring the movement of data from the inputs to the outputs by way of a data counter. In contrast, *reconfigurable computing* is based on a data-stream paradigm where the data-path is optimised for a specific processing task, allowing the focus to be on the data flow, rather then the instruction flow. Being able to manipulate the data path as well as the control-flow of a computing system means that we can optimise the system for almost every type of computing problem, while at the same time replicating the individual operations across a *reconfigurable fabric* or *reconfigurable array*. This process is of course limited by the size of the array in question, as synchronisation between reconfigurable units in a large system faces the same original dependency issues as existing multiple processor systems.

The most common form of reconfigurable fabric today is an FPGA, although the concepts explained here could just as well be applied to other programmable logic technologies. In essence, an FPGA is a commodity-priced device that has an array of thousands of logic gates that can be reprogrammed to implement a

limitless variety of logical circuits. This means they can be programmed to tackle all sorts of parallel computational tasks. At the same time, by implementing so much customised parallel logic in a single silicon device, there are large improvements in power consumption over CPUs, sometimes up to 70%. Ultimately, if implemented properly, the customisation of FPGAs allows a better usable logic density, resulting in reduced chip sizes and hence cost for a given computing task [11], compared with CPUs. Their performance can clearly be beaten by custom ASIC designs, although the costs of ASIC design and fabrication can be enormous[1], and ASICs lack the flexibility of being reconfigurable.

Whilst FPGAs were originally expensive with only a relatively minute amount of reconfigurable logic, they have progressed in line with the advancements in silicon fabrication, and the associated increase in chip transistor counts. Thus, FPGAs are now resourceful enough to implement entire systems. This is facilitated by on-board hard-wired low latency RAM, and high speed I/O capabilities. At the same time, due mostly to economies of scale, the price of FPGAs in terms of programmable gates is falling [12].

## 2.4 Reconfigurable Co-Processing

Whilst there has been significant advancements in reconfigurable computing (see the likes of *Cray*, *SGI*, *SRC Computers*, *Celoxica* and *Nallatech*), the field is by no means fully explored. In many respects, we could say that this is just the beginning for the use of reconfigurable technology as an approach to high performance computing, as currently there is not much reconfigurability to be seen in the various computing centres around the world. With regard to the general purpose CPU, it still plays an important role in reconfigurable systems, as a means to execute non-parallel code quickly, as well as to provide an easy platform on which to run an operating system with existing software resources. The known approach is to add reconfigurability by means of a *reconfigurable co-processor*. In this configuration, an FPGA acts as a co-processor to the general purpose CPU, in much the same way as a 3D graphics card acts as a co-processor to gaming machines to add acceleration for various graphics routines. Figure 2.2 demonstrates how we would want to incorporate a reconfigurable co-processor into a computing system. By giving the co-processor direct access to the primary memory as well as I/O resources, we can hope to build an extremely efficient and flexible computing system. The simple rule of thumb, is to execute as much of the sequential procedures as possible on the CPU, and as much of the parallel procedures as possible on the co-processor.

In real world scenarios, minimising latency between the co-processor and the CPU is extremely important as any time gained by executing functions on the co-

[1]There are businesses whose sole purpose is to estimate the cost of custom ASIC designs. See `http://www.chipestimate.com` as an example.

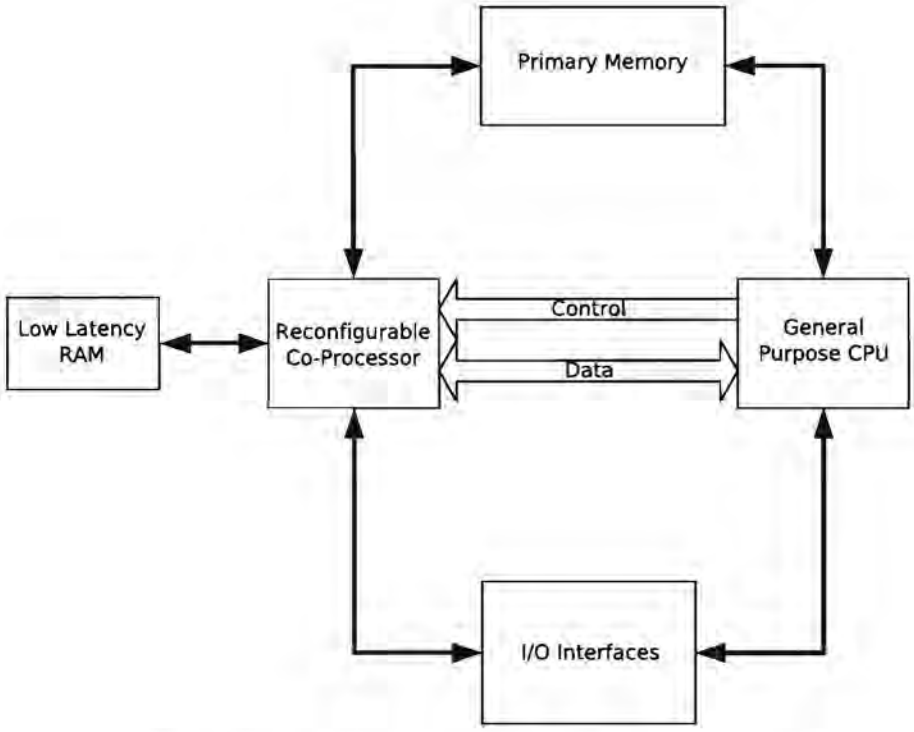



Figure 2.2: A functional view of a reconfigurable co-processor

processor can easily be nullified by the time lost transferring data to and from the co-processor [13]. To supplement possible latency issues, we require additional low-latency memory close to the FPGA. At the same time, no amount of available I/O bandwidth should be spared, as we wish to make full use of the FPGAs resources for less compute intensive operations with heavy data streams. By using an FPGA in this configuration, we can also allow the computing system to be entirely data-stream-based where the CPU is merely a auxiliary component which monitors and controls the processing, and there is no significant data flow between CPU and co-processor. However, this would only suit certain data-stream based applications such as digital signal processing. By giving the co-processor direct access to high speed I/O, we can also hope to create a cluster of reconfigurable nodes (see Section 4.2.1).

Additionally, in order to make the the system as flexible as possible, we require it to be dynamically reconfigurable [14]. This implies that we can redefine the FPGA during application run-time to target it to a new processing task, thereby leveraging the FPGA's resources for a range of particular computing problems within an application. Current FPGAs can be reprogrammed with new bit-streams in a matter of seconds, allowing them to tackle a myriad of different computing problems[2], as long as their external hardware interfaces are flexible and resourceful enough to do so. As a result of the above criteria, a convenient way of including a reconfigurable co-processor is by assembling it onto a separate daughter-card (known from hereon as an *accelerator card*) with dedicated RAM and I/O resources, which is then in-

[2]for a comprehensive list of problem types see [15]

terfaced with the main host machine via a standard transport bus. This practical approach to a reconfigurable co-processor is demonstrated in Figure 2.3.

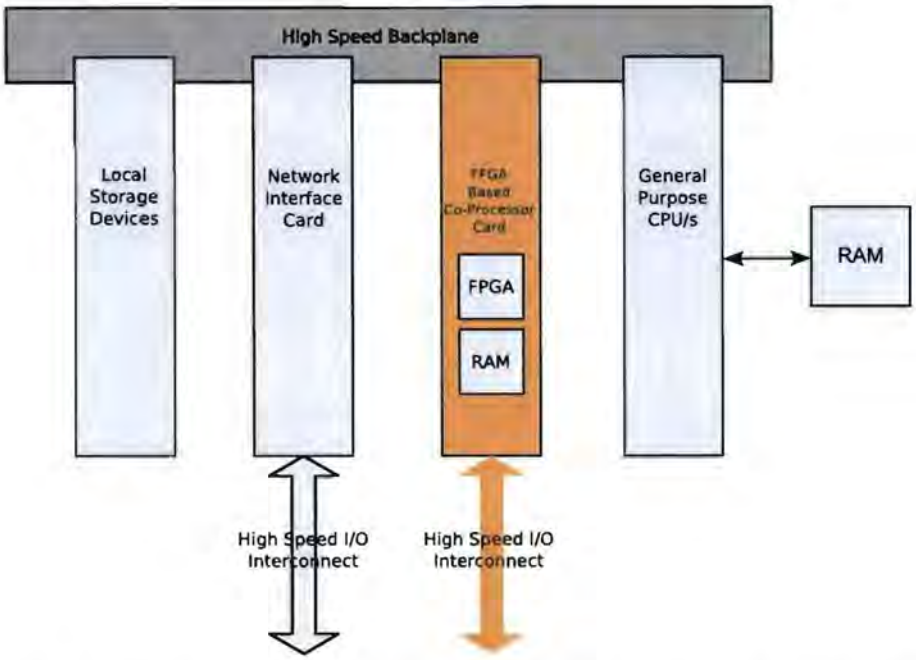



Figure 2.3: A practical view of a reconfigurable co-processor interfaced with an existing Von Neumann node

# Chapter 3

# A Technology Review

This chapter describes the investigation carried out into the existing technologies that are applicable to a reconfigurable accelerator card.



## 3.1 Existing Accelerator Cards

### 3.1.1 Celoxica's RCHTX-XV4

This successful accelerator card is designed to be used in AMD Opteron systems [16]. It uses the HyperTransport Extension (HTX) standard to allow it to connect to a host system. The card contains two Xilinx Virtex-4 FPGAs, one of which acts as a "*kernel*" device which is used to implement a HyperTransport interface to the host machine, and to allow dynamic reconfiguration of the other "*user*" FPGA. The *kernel* FPGA is interfaced with a flash storage device that provides non-volatile storage for bit-streams and software files. Each FPGA is interfaced to multiple QDR memory devices which serve as high speed local SRAM storage space. The excess pins available on the *user* FPGA are made available via a LVDS expansion port. For flexibility, the card also has a VGA/DVI port and a Gigabit-Ethernet port (see Figure 3.1).

Celoxica currently advertises the card as an accelerator for financial instrument calculations, oil and gas and life sciences, but there is no reason for it to be limited to these applications. It is provided with various APIs for uploading bit-streams and for transferring data to/from the card's memory. There are also various HDL controller cores that come preset for the cards interfaces.

### 3.1.2 University of Mannheim HTX-Board

Another AMD Opteron targeted accelerator card is the HTX-Board [1]. Its core component is a Virtex-4 FX60 FPGA that interfaces to a variety of useful devices. The card makes use of the HTX standard, but unlike the RCHTX-XV4 card, its HyperTransport interface and user logic are implemented on the same FPGA. This

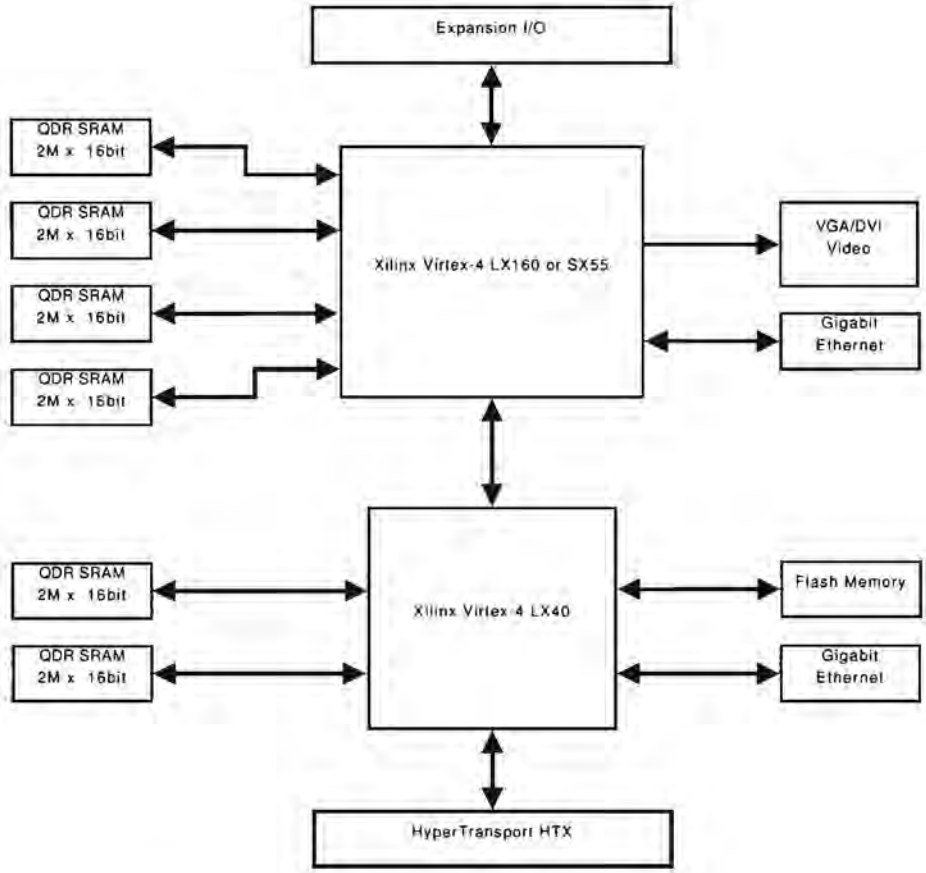



Figure 3.1: Celoxica RCHTX-XV4 architecture

will inevitably lead to greater usage restrictions for that FPGA, since the controller core takes up logic space. The card also contains between 128 and 512Mbytes of DDR2 DRAM, a large amount of on-board storage compared to other cards. Functional utilities include a Gigabit-Ethernet connector, USB port, 6 SFP sockets which can be driven using the Virtex-4's built in multi-gigabit transceivers, and a 512Mb FLASH storage device. There is also a mezzanine connector to allow multiple cards, or other suitable peripherals to be connected together via high speed LVDS signalling tracks (see Figure 3.2).

The Virtex-4 FX60 contains two hard-wired PowerPC cores which can be used to turn the card into a standalone embedded platform, should these be needed. The purpose of the HTX-Board is to showcase the HTX standard and to be a test-bed for custom designs that make use of this standard. The card was produced by the University of Mannheim's Computer Architecture Group and is open-source. In addition to this, open-source HyperTransport controller cores written in Verilog can be downloaded from the groups website[1]. The group has recently been taken over by the University of Heidelberg.

[1] `http://www.htce.uni-hd.de`

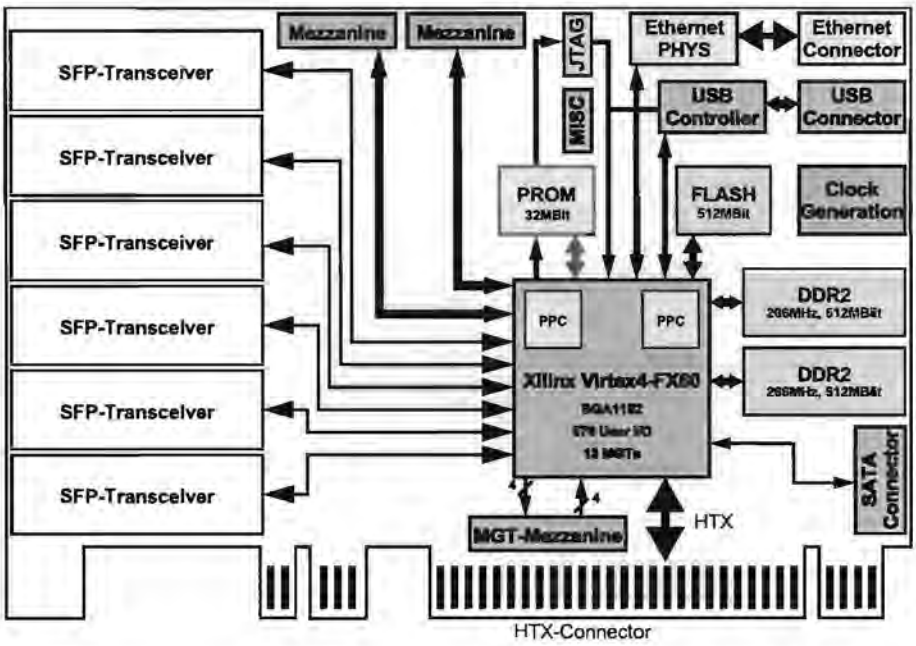



Figure 3.2: University of Mannheim's HTX-Board architecture, from [1]

### 3.1.3 Nallatech BenONE-PCIe

This card is based upon an 8-lane PCI Express 1.1 standard for connecting the expansion card to the host system [17]. The card incorporates a Virtex-5 LX50T FPGA that is interfaced to two QDR-II SRAM banks, as well as the necessary JTAG and non-volatile PROM memory (see Figure 3.3). Most importantly the card is designed to provide expansion capabilities though a custom DIME-II Module Expansion slot, allowing various mixed-signal or digital circuits to add processing and I/O capabilities. Nallatech also provides the *DIMEtalk* software suite which supports design flows where various functional blocks such as interface and memory controllers can be incorporated with user logic blocks in a GUI-type environment.

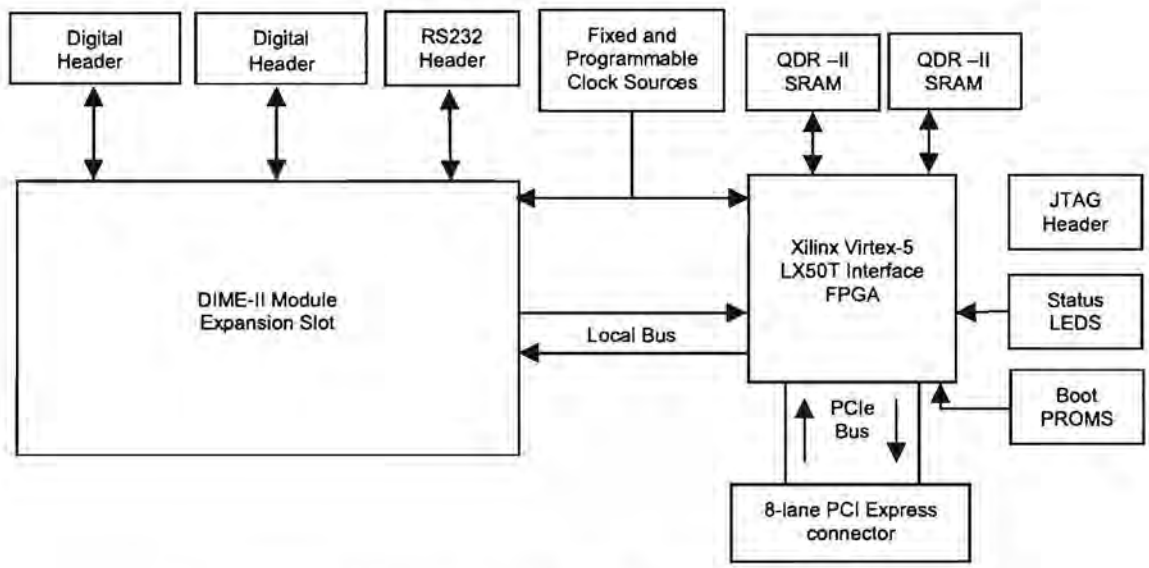



Figure 3.3: Nallatech BenONE-PCIe architecture, from [17]

### 3.1.4 Pico Computing E-16 LX50

The previously discussed cards have all been a similar form factor; that is they are all adaptor cards that are meant to be plugged into a host motherboard. The E-16 LX50 offers the simple architecture of a single Virtex-5 LX50 FPGA connected via a 1-lane PCI Express bridge to the host laptop machine (see Figure 3.4). To do this it uses the *ExpressCard* form factor standard. The card also contains two PSRAM banks offering 32MB of storage, as well as an internal JTAG chain. Software utilities include *PicoUtil* (Pico's proprietary FPGA management software), as well as *ImpulseC's CoDeveloper Platform* for C-to-HDL compilations. This card has limited power supply because of the limitations of the ExpressCard standard and thus the LX50 cannot be fully utilised, a limitation which may or may not affect a given application. Physical design of this device was clearly more difficult and constrained by its very slim form factor.

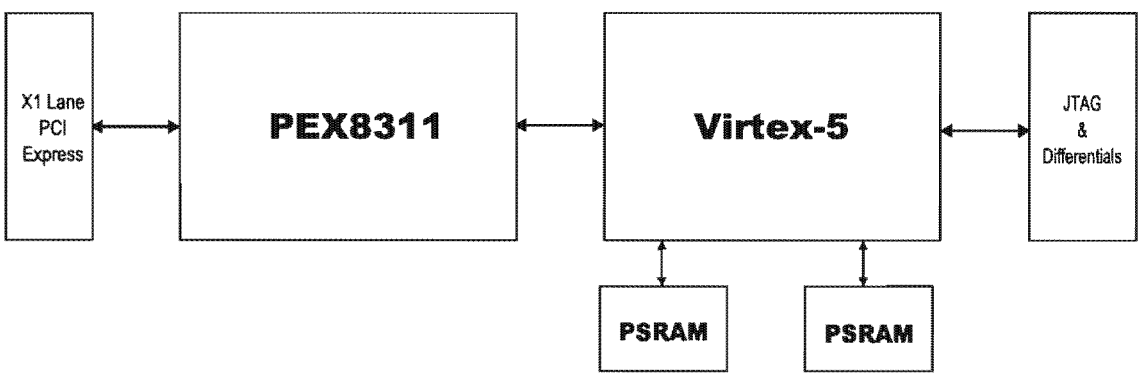



Figure 3.4: Pico Computing E16 architecture, from [18]

## 3.2 Top-End FPGA Technology

**Xilinx**[2] and **Altera**[3] are the two leading producers of high performance FPGA devices. Currently Xilinx produces its Virtex-5 range of FPGAs with the largest specimen (the Virtex-5 LX330T-FF11738) having approximately 330,000 logic cells. Altera's top of the range Stratix-IV FPGA has 640,000 logic cells. This is by no means their only measure of performance. Each FPGA comes with it's own type of on-board static RAM, high speed serial transceivers, hardwired multipliers, IO Buffers, digital clock management circuits, and various hard-wired ASIC-type units such as PCI Express endpoints and embedded processors such as the PowerPC. Pin-outs range from approx 300 pins to more then 1700 pins on the largest devices, giving them huge I/O capabilities. It's important to note the the high speed transceivers built into the latest FPGAs are a primary reason why they have become so popular to design with, as this reduces PCB complexity drastically.

[2] `http://www.xilinx.com`
[3] `http://www.altera.com`

Another noteworthy producer of FPGA products is **Lattice Semiconductor**[4].During the course of writing this thesis, Lattice have started to produce a high performance range of FPGA devices that compete favourably with the lower end Stratix-IV and Virtex-5 devices, especially considering their reduced cost (as much as 4 times cheaper[5]).

Each producer of FPGAs ships it's own compilation suite to allow generation of bit-streams to configure their devices. The pros and cons of the different offerings often prove an important decision when selecting which manufacturer to use.

## 3.3 Expansion Card Interfaces

There are currently two competing standards for expansion cards that are suitable for high performance computing[6]: HyperTransport Extension (HTX) and PCI-Express. Both support multiple clock rates, and have their advantages and disadvantages as shown below.

### 3.3.1 HTX vs PCI Express

The **HyperTransport** standard is used predominately by the processor manufacturer AMD as a high-speed parallel chip-to-chip interconnect standard for it's Opteron CPU range. The specification allows for 32 bit links running at speeds ranging from 200MHz to 2.6GHz [19]. The standard's governing body, the *HyperTransport Consortium*[7], specifies an extension standard known as HyperTransport Extension (a.k.a. HTX) [20] which is a 16-lane[8] expansion card connection on a AMD Opteron based motherboard that consists of a 1x and a 16x PCI Express socket (arranged in reverse to avoid confusion). This link offers clock rates of 800 MHz (DDR) and couples the external card into the host's HyperTransport chain - essentially including it in the front side bus. This allows an HTX-based expansion card to have direct access to the host CPUs and system DRAM (via the CPU memory controllers). Figure 3.5 demonstrates this concept. HyperTransport has benefits in improved throughput compared to alternative approaches such as PCI,PCI-X and improved latency over PCI Express whose links operate on a secondary bus, and require bridging to the CPU/RAM Front Side Bus.

HyperTransport is considered a mixed serial/parallel link since it has numerous lanes and but is also driven at a high clock rate (400MHz+), giving it both low latency and high throughput. Data signals and control signals travel down separate

[4] `http://www.latticesemi.com`

[5] According to prices from distributor *Avnet Electronics*, and the Lattice's online store `http://www.latticestore.com`

[6] The older PCI and PCI-X technologies do not offer nearly the same throughput as these two newer standards and as such have been neglected.

[7] visit `http://www.hypertransport.org`

[8] Here 16-lanes refers to 16 lanes in either direction (i.e. 32 data signals).

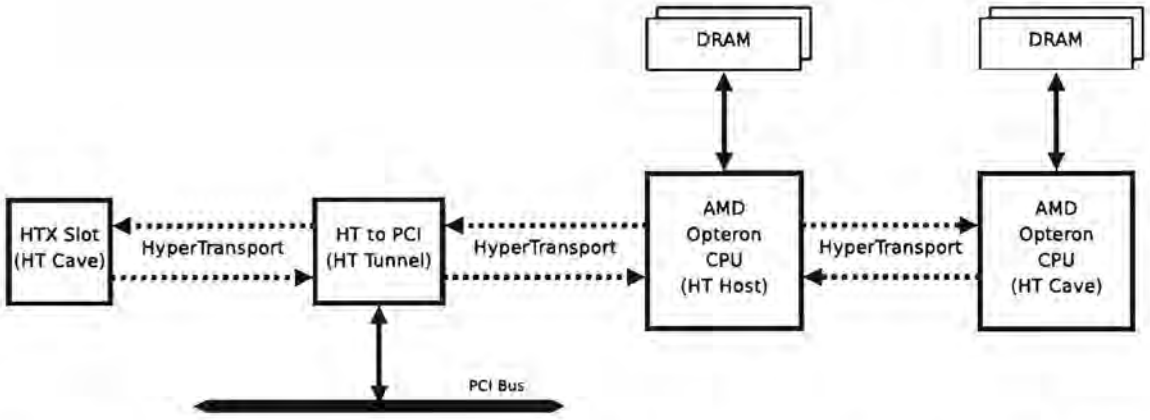



Figure 3.5: A Simple HyperTransport Chain with multiple CPUs and an HTX expansion slot

tracks. Data is packetised during transmission but with small packet headers (< 12 bytes ) relative to packet size. The protocol allows control packets to be inserted arbitrarily during data transmission to help decrease latency for various operations.

The HTX standard is not widely supported in industry, and there are few expansion cards that make use of the standard, although as can be noted from 3.1.1 and 3.1.2 there are cards which make use of it's lower latency for acceleration using FPGAs.

**PCI Express** is a high speed serial point-to-point link for expansion cards that uses 2.5GHz serial lanes to transfer data sequentially [21]. Links of 1, 4, 8 and 16 lanes are supported, and cards can be used in any slot supporting the required number of lanes or more - allowing much scalability. Unlike HyperTransport, which does not use encoding unless it is needed, PCIe uses by default a 8b/10b encoding scheme to incorporate data and timing in one signal. Each individual byte is sent down successive lanes and using complex synchronisation the bytes are reordered at the end point. This means that individual tracks do not have to be physically length-matched precisely - a great design benefit. However the reordering of bytes does cause an additional latency penalty. The PCI Express bus does not link directly with the host CPU, but is bridged into the Front Side Bus via a PCI Express capable chipset on the motherboard which acts like a switch for the different PCI Express links in the system (see Figure 3.6). As such, a PCI Express card does not have direct access the system DRAM or CPU. It can be shown that a combination of the increased packet overhead, the switching via a chipset and the 8b/10b encoding all result in an increase in latency compared to HyperTransport [13]. Table 3.1 shows a comparison between the two standards. These are the one-way MPI latencies on the two variations of *Pathscale's InfiniPath InfiniBand Network Interface* cards, the *QLE6140* and the *QHT6140*. The PCI Express version is a 8-lane device, but due to the tiny packet size this is deemed to be of little consequence. Although not very scientific, they are the best available comparison between the two standards in terms of latency, and indicates that HTX is offering between 300 and 150ns improvement

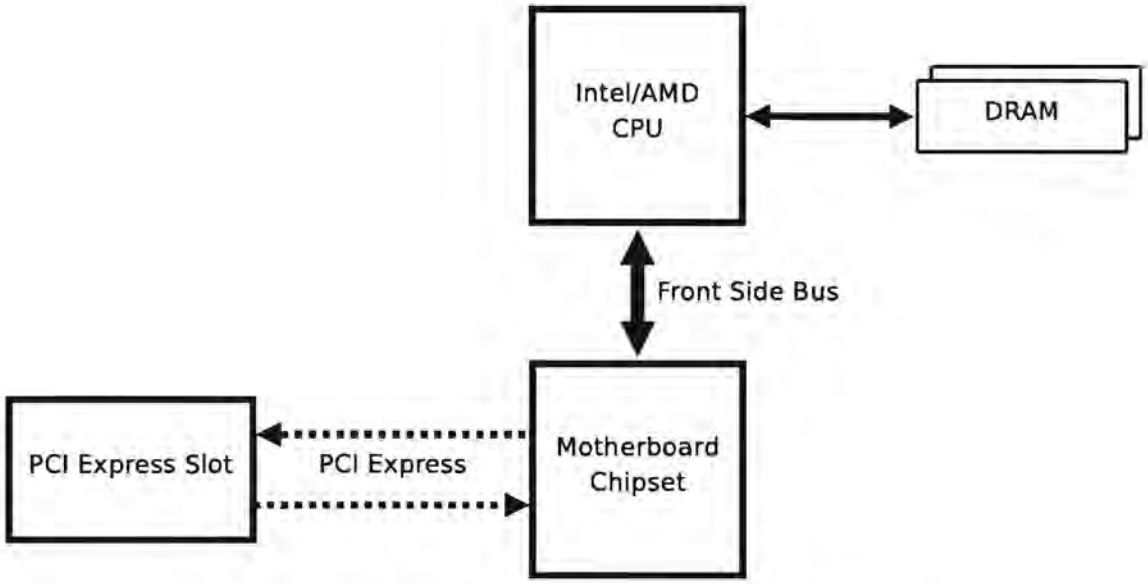



Figure 3.6: Simple PCI Express interface to the front side bus

in latency between two NICs.

Latency aside, PCI Express is already widely supported in industry. In fact, even AMD themselves offer bridge chipsets from their HT front side bus to a PCI Express controller switch[9]. As such, there are many more host systems to choose from that support the PCI Express standard.

| | **HTX 1.1** | **PCI Express 1.1** |
|---|---|---|
| *Clock Rate* | 800MHz | 2.5GHz |
| *Max Aggregate Throughput (16-lanes)* | 6.4 GB/s | 8GB/s |
| *Latency Comparison* | 1.29 $\mu$s | 1.6 $\mu$s |

Table 3.1: Comparison between HTX and PCI Express expansion standards

## 3.4 On-board Memory Technologies

Random Access Memory generally comes to in two distinct flavours: RAM with density and high latency or RAM with low density and low latency. From the cards discussed in Section 3.1, we can see they generally choose one of the two types of memory. The following lines give descriptions of the various *available* memory technologies.

**PSRAM** Pseudostatic RAM: A form of high density DRAM that "acts" like SRAM and has built in refresh functionality. It is generally regarded as a low power low performance alternative to other SRAM technologies.

**DDR-SDRAM** Double Data Rate Synchronous Dynamic RAM: a common form of memory where data is transferred twice every clock cycle on a single bi-directional 64-bit data bus. It comes in 3 generations (from DDR1 to DDR3

[9] An example of this is the AMD 770/790 chipsets

) with effective speeds of 200MHz to 1600MHz. Latency is seen as an issue here as it does not scale with increased frequency, and in fact in many cases increases due to constant CAS[10] latencies.

**QDR-SDRAM** Quad Data Rate SRAM: a popular version of high speed SRAM which is low density but very low latency SRAM modules. There are separate read and write data buses of varying width, and data is transferred on the rising and falling clock edges - hence the quad data rate. Simultaneous write latencies of 1 clock cycle and reads of 2 clock cycles are available with QDRII memory - the latest variety, with clock rates of up to 450 MHz. Densities range up to 72Mbits, meaning QDR is well suited to applications requiring low latency, at the cost of density.

**XDR-DRAM** Extreme Data Rate DRAM: A very low latency form of DRAM, with latencies as low as 1.25 ns, however there is a narrow (16-bits or less) bidirectional read/write data bus. It is not widely available.

**ZBT-SRAM** Zero Bus Turnaround SRAM: Another low latency SRAM with a single data bus, however there is no delay between switching between read and write operations. It is not widely available.

High density RAM is generally used to store large data sets, where the application is more data intensive then computationally intensive. Low density RAM is generally used to provide high speed access to small data sets where the application is more computationally intensive and less data intensive.

## 3.5 Back-end I/O Capabilities

Most of the existing accelerator cards make use of one or more high speed I/O interfaces, to expand the functionality and flexibility of the card. An example of where this could be used is in astronomy where data could be piped in to the card directly from an external source, processed on the card and the resultant data transmitted to the host machine. There are of course many situations where the I/O ports could be used.

**InfiniBand** is a switched fabric networking standard that is used widely in high performance computing systems to connect processor nodes to I/O nodes. The standard supports fibre as well as copper mediums. There are various data rates depending on link width, although copper 4-lane links at 2.5 Gb/s per lane are common. It is proprietary and thus like most closed standards is difficult to implement freely without dealing with royalties and other legalities.

[10]This refers to Column Address Select timings.

**10-Gigabit-Ethernet** is an open standard ratified by the IEEE which supports both fibre and copper links, the most common at this time being the Copper 10GBASE-CX4 standard. This consists of relatively low-cost 4-lane InfiniBand style connectors where each lane is run at 2.5Gb/s. Copper CX4 links are limited to 15m. The fibre versions (10GBASE-R) of the standard are more expensive requiring optical transceivers modules. There are great variations in the fibre implementations of 10GbE, for more information see [22].

## 3.6 Power Supply

Both the PCI-express and HTX standards offer 12V and 3.3V supplies for expansion cards using up to 75W of power. Smooth voltage supplies are then delivered to components through a combination of on-board switched and linear voltage regulators. An exception is the Pico Computing E-16 LX50 which (being a different form factor) gets its supply through the supply pins of the ExpressCard connector, and cannot deliver the full power requirements of its LX50 FPGA for intensive applications - a fairly undesirable limitation.

# Chapter 4

# The Requirements of a General Reconfigurable Co-Processor Card

In this chapter we discuss the project requirements in light of the technologies available, and the discussions in Chapter 3.

## 4.1 Form Factor

In keeping with the tradition of existing cards, it is worth noting the following benefits of a design using an expansion card for ATX/BTX form factor:

- Ease of installation
- Close coupling with host CPU and system RAM
- Access to rear I/O panel of the host system using a steel bracket
- Sufficient power supply through the expansion slot and/or directly from host PSU
- Relaxed maximum dimensions (compared to mini standards such as ExpressCard)

Thus it would be desirable to base the accelerator on either a HTX or PCIe expansion card standard.

## 4.2 Functional Requirements

### 4.2.1 Usability in Different Architectures

In order to make the card as flexible as possible, we would like to use it in one of the following configurations:

1. an accelerator for a standalone host machine (Figure 4.1) - referred to here as a single *reconfigurable node*.

2. a cluster of reconfigurable nodes (Figure 4.2)

3. an array of standalone cards (Figure 4.3)

In use case 3, the cards would need power to be provided by some external supply unit, as they will not be powered via their expansion interfaces. For the sake of simplicity, the networks shown have been simplified greatly. In practice, the data storage system and administration would be more complex.

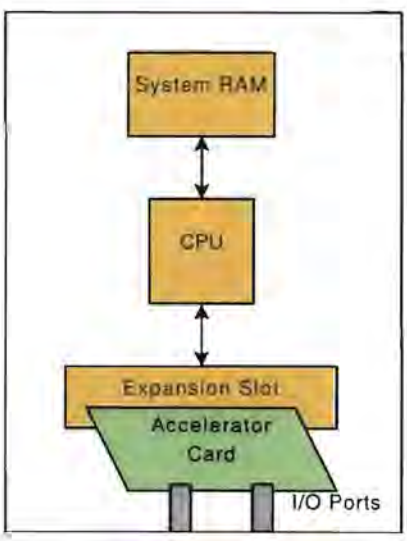



Figure 4.1: Use case 1 - Single host machine

As we can see from the use cases, the card will require both a expansion interface to a host node, as well as high speed I/O ports for networking and flexibility reasons.

### 4.2.2 I/O Capabilities

For applications that require large data rates, we require the latest generation networking standard. For the purposes of an open design, 10Gigabit-Ethernet remains the most sensible choice, thus the card should have at least one 10GbE connection.

### 4.2.3 Reconfigurable Component

In Section 3.2 we discussed the latest FPGA technology. It is clear that the card requires either a Xilinx Virtex-5 or Altera Stratix-IV FPGA to provide our next generation reconfigurability. We would like (within reason of availability and cost) the most reconfigurable logic resources as possible, as the needs of any user applications are not known. The chosen FPGA must also support the I/O requirements discussed in 4.2.2.

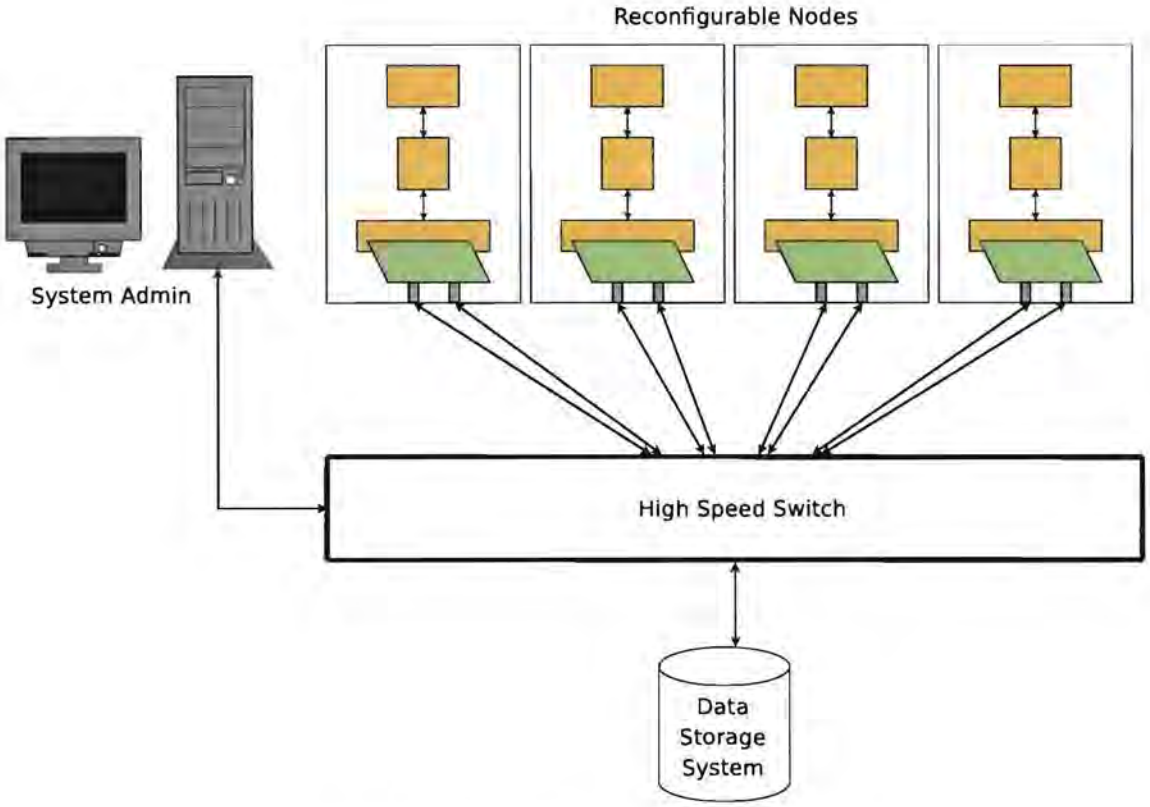



Figure 4.2: Use case 2 - Cluster configuration

### 4.2.4 Latency

In high performance computing, latency is an important factor in improving performance when dealing with multiple compute nodes working in parallel. For the purposes of acceleration using a co-processor, latency is of utmost importance (see [13]). Since we lack a specific application's latency requirement, we can safely assume that we want I/O interfaces and on-board memory devices that can deliver the lowest latency possible.

### 4.2.5 On-board Memory

It's clear from Section 3.4 that SRAM still outperforms DRAM in terms of latency. The accelerator card requires high speed, low latency SRAM as its core memory technology. This is because a high throughput, low latency interface to the host node using either PCIe or HTX allows the accelerator to use the system RAM for large data sets, leaving the data for immediate use in the on-board SRAM modules.

### 4.2.6 Power Consumption

Within reason, the card should be able to run within a host node using the host nodes power supply. This means there is a maximum limit of 75W (the maximum power supply specified by the PCIe and HTX standards). Separate power connectors generally supply more then the expansion slots pins so this is not an issue. For

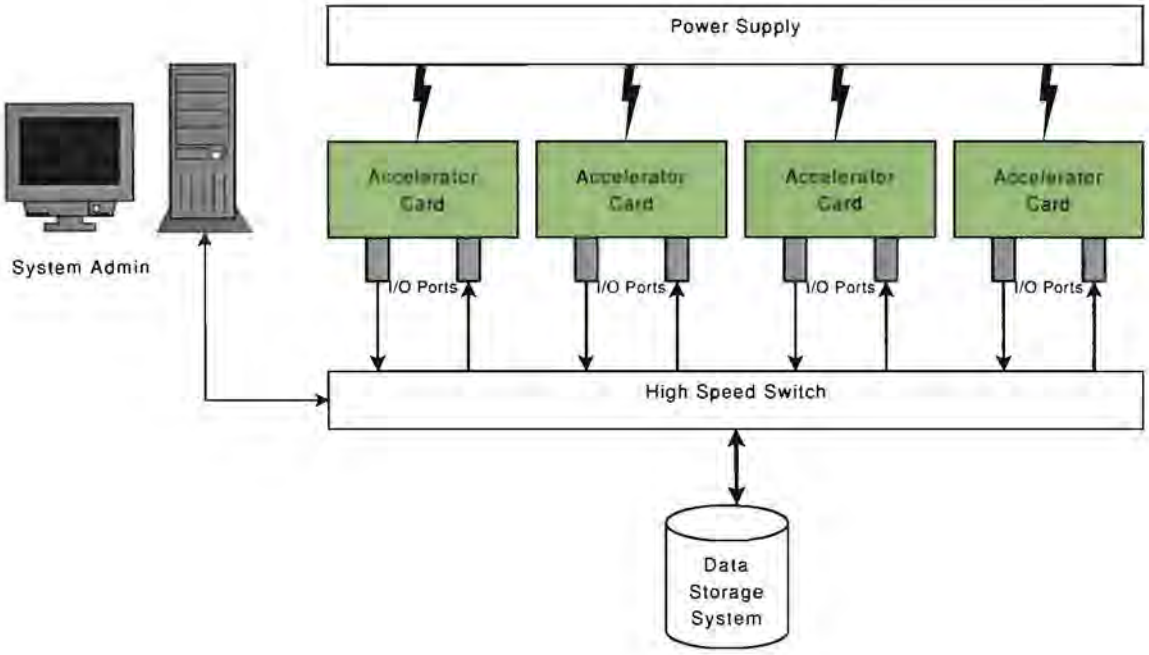



Figure 4.3: Use case 3 - Cluster configuration (stand-alone)

the purposes of operating the card within 1U rack spaces, there is also a thermal dissipation requirement, although for the purposes of this project, this will not be considered[1].

## 4.3 Cost Considerations

FPGA devices are still relatively expensive and the NRE costs of producing an accelerator card are significant. For example, the cost of Nallatech's older generation H1010-PCIXM is currently ZAR 52,000 excluding warranty, support costs and software tools. Newer generation devices would almost certainly cost more. The cost of producing a piece of educational hardware such as the one in this project is difficult to calculate and compare to commercial products owing to the completely different cost environments. Benefits of the academic environment are student salaries, relaxed quality standards and regulations, free tools and donations. Disadvantages are increased design time and low volumes - a vitally important factor in the electronics industry. As such, low cost is considered to be neither a significant nor relevant requirement for this project.

[1]It is not foreseen the the card will be used in a rack server environment during it's first revision.

# Chapter 5

# Hardware Design

This chapter details the decision making that went into producing the conceptual hardware design. The inclusion of each component and interface is explained in light of its capabilities and the requirements specified in Chapter 4.

## 5.1 Subsystems

### 5.1.1 FPGAs

As it turns out, compared to Altera, Xilinx offers better support for their products within South Africa, and their devices are used extensively by the Karoo Array Telescope Project[1] (the author's bursars) and their collaborators, the CASPER[2] group at the University of California, Berkeley. In addition, even though both manufacturers offer donation programs, Xilinx was willing to donate eight valuable FPGA devices. They agreed to provide four of each of the Virtex-5 LX110T-FF1136 and the smaller Virtex-5 LX50-FF676. Since the FPGAs are by far the most costly components on the card, and that these were the latest generation FPGAs, it was hugely beneficial to have these devices donated - and so both[3] devices were included in the design. Table 5.1 on page 28 illustrates the capabilities of the two FPGAs.

| | LX50-FF676 | LX110T-FF676 |
|---|---|---|
| I/O Pins | 440 | 640 |
| Logic Cells | 46,080 | 110,592 |
| Block RAM (Kbits) | 1,728 | 5,328 |
| Digital Clock Managers | 12 | 12 |
| RocketIO GTPs | 0 | 12 |
| DSP48E Slices | 48 | 64 |
| Max Clock Rate (MHz) | 550 | 550 |

Table 5.1: Specifications for the on-board FPGAs [23]

[1] visit `http://www.ska.ac.za`

[2] For more information visit `http://casper.berkeley.edu`

[3] The reasons for having two FPGAs on the card are explained in 5.1.2

### 5.1.2 FPGA Configuration

In Celoxica's design of the RCHTX-XV4 (see Section 3.1.1), two FPGAs were used: one smaller device to control the host machine interface, and one larger, more powerful FPGA to run the application. This configuration allows the RCHTX's main FPGA to be reconfigured on the fly whilst keeping the host HyperTransport link alive. It would be clearly problematic to reconfigure a single FPGA when the HyperTransport interface is up and running as this would require partial reconfiguration. This configuration also saves the additional logic cells the HT core would use if it were implemented on the same FPGA as the user gateware, allowing the excess logic on the smaller LX50 to be made available for user gateware. Figure 5.1 demonstrates how the two FPGAs are arranged relative to the HTX interface.

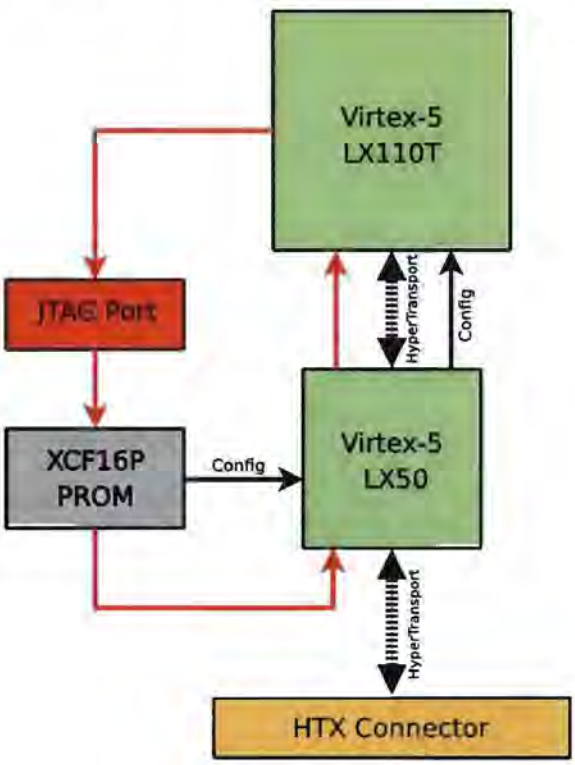



Figure 5.1: The multi-FPGA setup with HyperTransport links and configuration options

**PROM** The Virtex-5 FPGAs support a range of configuration options. In order to store a default bit-stream for the LX50 we require some non-volatile storage from which the FPGA can fetch a bit-stream upon power-up. Xilinx offers PROM devices which implement a simple configuration interface which the Virtex FPGAs support. The smallest device capable of storing a LX50 bit-stream is the XCF16P PROM device. The bit-stream for the LX110T can then be loaded from the host node and programmed into the LX110T using the LX50. There is a programming interface between the LX110T and LX50 similar to the one linking the XCF16P and the LX50.

**JTAG** An on-board JTAG port is vital for initialising the PROM device, as well as debugging the board. Initially the JTAG port will be used to program both FPGAs directly through the JTAG chain using a Xilinx USB programming cable. Once the bit-stream for the LX50 has been finalised, it can then be stored in the XCF16P PROM device to be loaded automatically into the LX50 upon power-up. The JTAG chain links all the programmable components allowing all of them to be accessed and configured via a single JTAG port.

### 5.1.3 HyperTransport Interface

HyperTransport has the lowest latency when compared to PCI Express [13]. Therefore it more closely meets the requirements of our reconfigurable accelerator card described in 4.2.4.

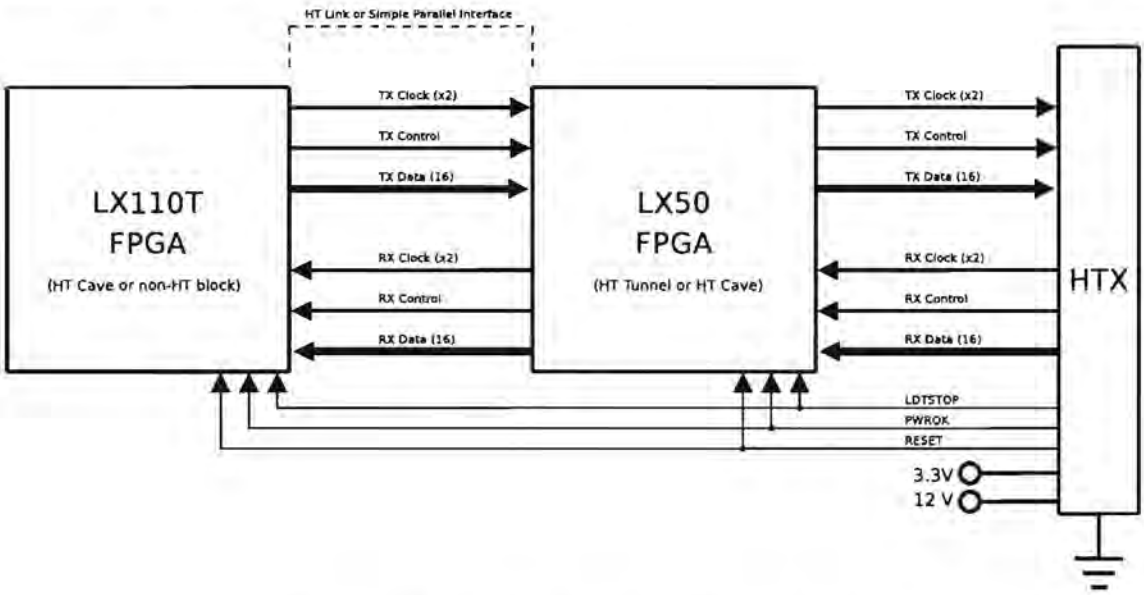



Figure 5.2: The on-board HTX interface

The HTX expansion slot is simply a standard connector that connects the HyperTransport tracks on the card to the appropriate tracks on the host motherboard. The version of HyperTransport specified for use over an HTX connector is Rev 1.03 which is considered in the latest HyperTransport (a.k.a HT) specification document [19] to be *Gen1* compliant (the latest HT specification is Gen3, offering higher frequencies). Thus the HT interface to be implemented on the accelerator card is Gen1. The Gen1 HT specification calls for 16 differential signalling pairs (2 bytes) in each direction. There are also two clock signal pairs (one for each byte) and one control signal pair in each direction. These form the high frequency signals which can are meant to operate at speeds of 400MHz DDR giving 800Mbps per differential pair. The use of multiple sub-GHz signalling pairs was an another advantage of HT over PCIe for the purposes of the card. PCIe would require multiple 2.5GHz signals which would mean requiring GTP ports on the FPGA - something the LX50 does

not have. There are also three other low frequency LVCMOS signals: PWROK, and RESET, LDTSTOP, all used to initialise or reset the HT link.

The two FPGAs are linked together by a continuation of the HT chain. The same signals which link the HTX connector and the LX50 are also implemented between the LX50 and the LX110T, as demonstrated in Figure 5.2. HT devices can either be *hosts*, *slaves*, or *tunnels*. By implementing the HT signals between the two FPGAs, we give ourselves the opportunity to either continue the HT chain into the LX110T by making the LX50 act as a HT tunnel device, or simply implement a straight parallel interface using the HT signals between the two FPGAs, meaning the LX50 would be a cave device (i.e. the end of the HT chain), and the LX110T would simply be an addressable processing block attached to the LX50 by the simple parallel interface. A simple parallel interface could be implemented over the physical HT tracks, since they are numerous (76 individual signalling tracks).

In addition to the data and control signals, the HTX connector also supplies 12V and 3.3V power pins which can be used to power the board. See section 5.2 for more information.

### 5.1.4 QDRII+ Memory

When designing the memory interfaces, we had to take into account the number of available FPGA pins there were to work with, as well as the type of high speed memory we could acquire. As it turned out, we were able to acquire donated QDRII+ memory devices from the SKA/KAT project, who kindly offered us their extra stock. These were Samsung K7S3218T4C devices which have a capacity of 2M memory cells each 18 bits wide. There are individual write (18 bits) and read ports (18 bits) and a single address port (19 bits). They use the single-ended HSTL-I 1.5V or 1.8V signalling standard and can operate at 400MHz. There was approximately enough physical space for six of these devices on the board without stretching the use of the FPGA pins: two memory modules for the LX50, and four for the LX110T. This worked out quite nicely in terms of board layout and pin usage, when we considered the needs of the HT interfaces, and the other I/O ports. Figure 5.3 shows a high level view of the QDRII+ interfaces.

### 5.1.5 10 Gigabit Ethernet

The choice between copper 10GBASE-CX4 and fibre implementations of 10GbE was an easy one. The optical transceivers for implementing 10GBASE-R are very expensive[4], and required additional space on the card for housing them. It also seemed wasteful to augment the use of the high speed serial transceivers (GTPs)

[4]Both XFP and SFP+ transceivers cost in the region of $500-$1000, according to prices on Ebay.com, as of November 2008. CX4 connectors cost in the region of $15. Fibre cables are also more expensive then copper CX4 cables.

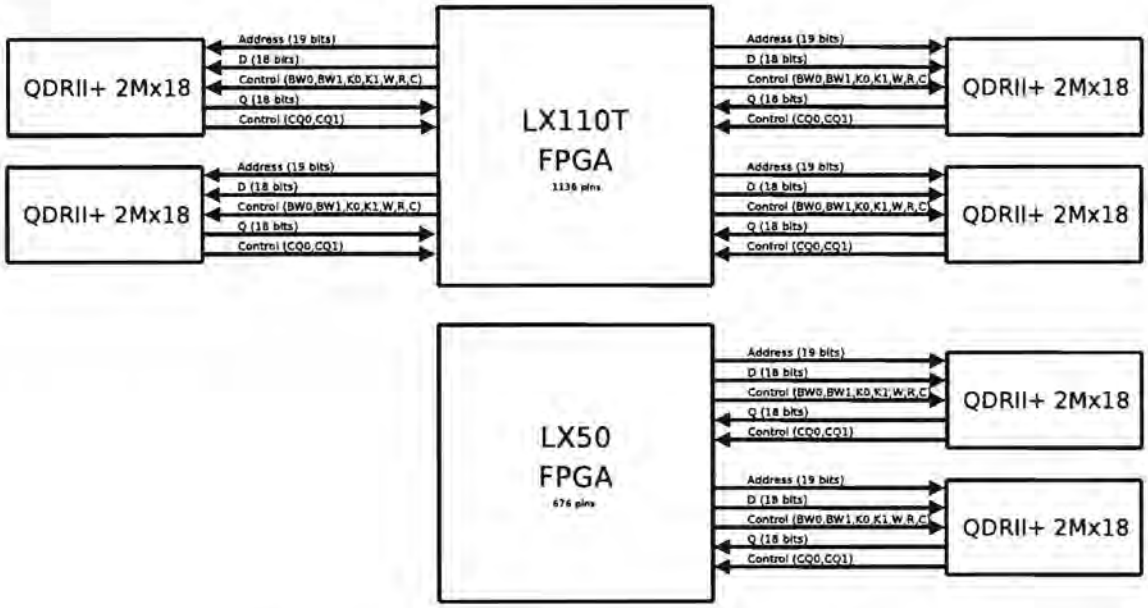



Figure 5.3: QDRII+ interfaces to the FPGAs

on the LX110T device, which were perfectly capable of implementing the 3.125 Gb/s signalling for the copper standard on their own, by using a XAUI interface between the FPGA and the CX4 connectors. Usually this type of setup would require a retimer PHY device close to the connector, however for the purposes of this project, the functionality without a retimer was adequate for the project[5]. Thus the GTP ports on the LX110T FPGA were setup to directly drive the 10GBASE-CX4 interfaces through a XAUI interface to the CX4 connectors (see Figure 5.4). The two 10GbE ports used half of the available GTP pins on the LX110T. So in addition to the usable ports, one set of GTPs was setup in a loop-back configuration in order to easily test potential controller cores without having a loopback CX4 cable attached.

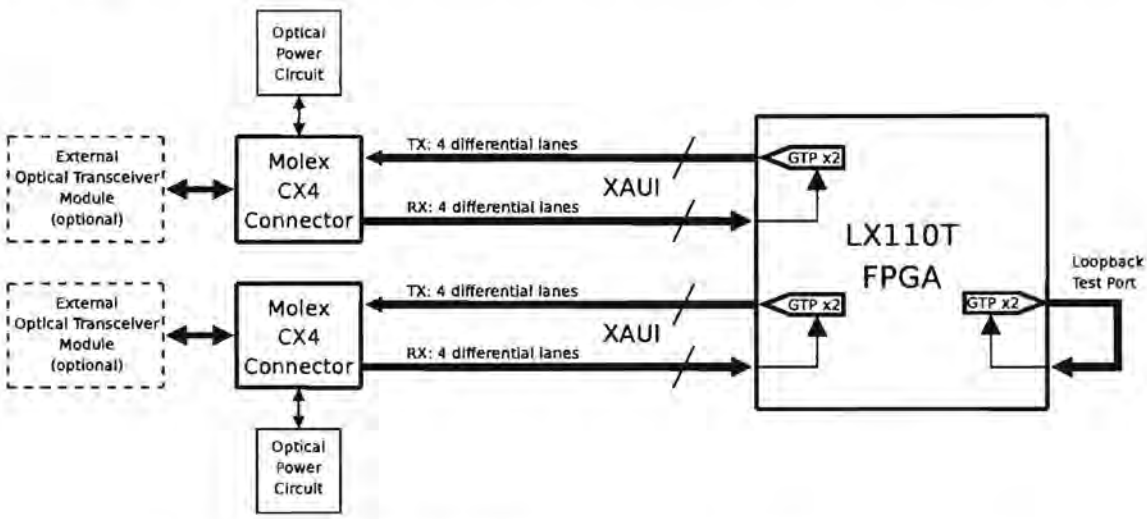



Figure 5.4: 10GBASE-CX4 interface to the CX4 connectors

[5]Without an on-board retimer PHY device, the LX110T could only drive short (~3m) CX4 cables - enough for our needs.

The 4-lane copper connectors (Molex 91804-0411) required for attaching the card to CX4 cables are also widely available and relatively cheap at approximately $16[6]. To add flexibility we added power circuitry to drive optical modules that could clip onto the CX4 connectors in an external manner. Fortunately the pins on the CX4 connector provide an easy way to do this by specifying particular ground pins to be multi-use. This additional circuitry consists of two voltage comparators, one AND gate and a 3.3V power distribution switch.

The use of the stringent signal integrity requirements, the GTP ports needed very stable linear voltage regulators, which will be discussed in more detail in Section 5.2.

### 5.1.6 Serial Debug

A serial debug port was implemented as in Figure 5.5 that connected the LX50 to a RS-232 transceiver (MAX3388ECUG), a PHY device which catered for dual send and receive ports. For debugging of the LX110T FPGA, these signals can quite easily be passed through the LX50 over the HT ports to the LX110T.

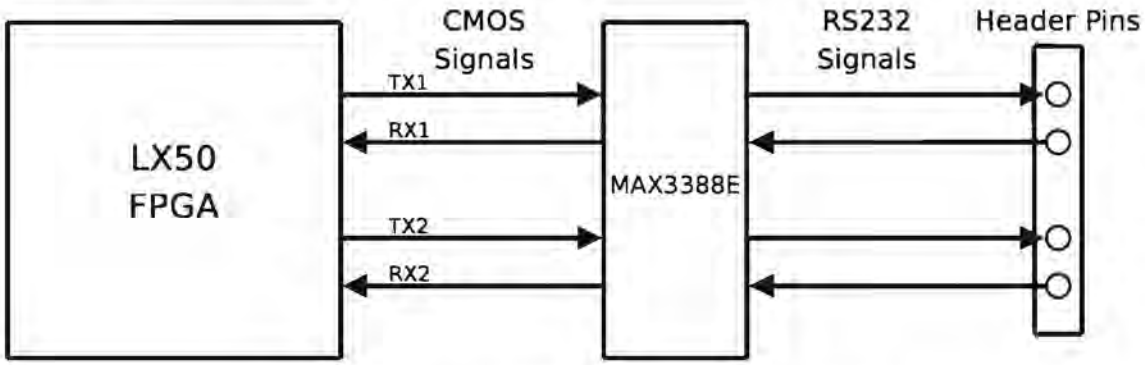



Figure 5.5: Serial RS232 interface to the LX50

### 5.1.7 LEDS

Each FPGA was directly interfaced with four green LEDS. There is also a LED to indicate that the main 12V-to-5V regulator is powered up and generating 5 volts.

### 5.1.8 Clocking

In order to generate stable clocks signals, the card required some crystal oscillators. Since there was no target application, two generic clocks were provided (100, 200 MHz). Each of these LVDS clock signals were then connected to 4-port SN65LVDS104 repeaters which produced multiple copies of the same clock signals, to be passed to different banks on the FPGA. This provided the flexibility of a single source clock to drive multiple parts of the FPGA.

[6]This is the cost of the Molex 91804-0411 from supplier Digikey at the time of writing.

The XAUI/10GbE controller cores required 156.25MHz reference clocks as part of the internal mechanism to generate valid 3.125Gbps signals [24]. These reference signals were generated using a single clock and then passed through the same repeater device as the 100 and 200 MHz clock signals, as demonstrated in Figure 5.6. The resulting duplicated clocks signals were then passed to the two 10GbE controllers, as well as to the single loop-back 10GbE controller.

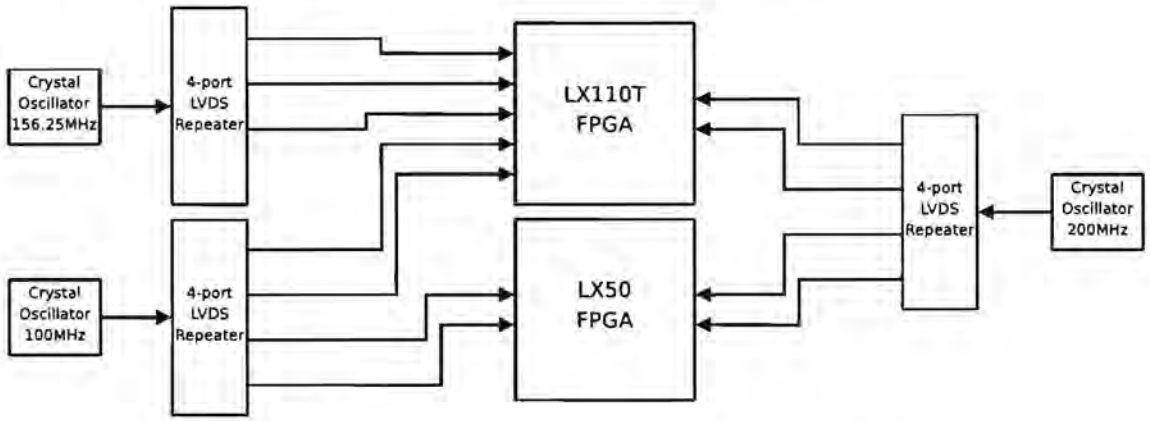



Figure 5.6: On-board clock generation using crystal oscillators

## 5.2 Power Supplies and Voltage References.

The estimated maximum power consumption of the card was calculated by adding up the ratings of all the ICs as well as the power consumption figures provided by a Xilinx FPGA power estimation spreadsheet[7]. The FPGAs each had numerous voltage and current requirements for different I/O banks, depending on which signalling standards were being used by the interfaces using those pins. Since different applications running at different clock rates would cause the FPGAs to demand variable amounts of power, we entered into the spreadsheet a worst case scenario with greater then 80% usage of all main blocks of the FPGAs, taking into account the designated I/O standards we wanted to implement (i.e. HSTL-I for Memory modules, HT for the HTX interface etc) as well as the possible user logic usage. The total power requirement was approximately 40W[8], however there were also various specific current requirements for the various voltages to provide for. Table 5.2 shows the calculated requirements for the different voltages on the card. Note that we provided for separate 1V supplies for each the FPGAs as this would provide greater voltage stability, and reduced load strain on each 1V regulator.

A network of power regulators was then specified for the required voltages and current loads. We used the Texas Instruments PTH08 range of regulators, as these were recommended by Xilinx for their FPGAs[9], and free samples were also provided

[7]see Appendix A- /power_and_thermals/
[8]see Appendix A- /power_and_thermals/
[9]TI PTH08 regulators were used in almost all of Xilinx's Virtex-5 development boards.

| Voltages | Required Current (mA) | Supplies to |
|---|---|---|
| 1 | 5653 | LX50 |
| 1 | 12886 | LX110T |
| 1.5 | 2992 | LX50, LX110T, QDRII+ |
| 1.8 | 5410 | QDRII+, XCF16P |
| 2.5 | 2207 | LX50, LX110T, QDRII+ |
| 3.3 | 405 | Crystal Oscillators, Repeaters |
| 1 | 228 | MGTAVCC |
| 1.2 | 117 | MGTAVCCPLL |
| 1.2 | 291 | MGTAVT |
| 0.75 (Ref V) | - | QDRII |

Table 5.2: Current requirements for the different voltage sources

through TI's sampling service.

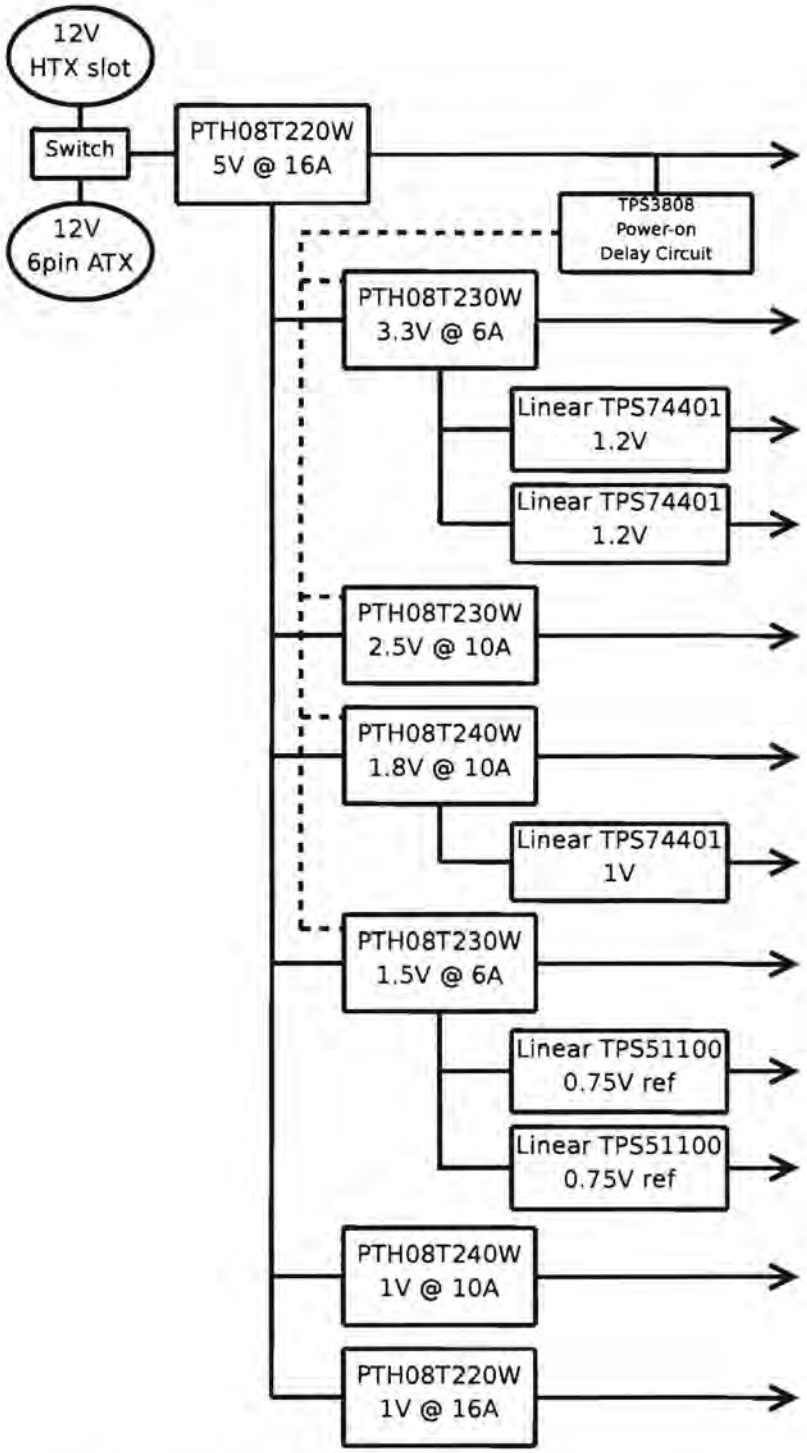



Figure 5.7: Power and reference voltage network

The 12V and 3.3V supply pins provided over the HTX connector provided great convenience for powering the card. To minimise current over the connector and to simplify the power network design, only the 12V supply was used. An alternative 12V supply was also made available via a 6-pin PCIe power connector - now a standard for providing power to expansion cards. The option of a 6-pin connector would allow the card to be powered when not plugged into a HTX slot. A simple jumper was used to select which 12V supply to use. The 12V was then fed into a 12V-to-5V step down regulator, which then supplied 5 volts to the other main regulators: 3.3V , 2.5V , 1.8V and dual 1V regulators. See Figure 5.7 for an illustration of the

regulator configuration. It was likely that switching on the 5V driven regulators simultaneously while powering up the 12V-to-5V regulator would cause a current spike that would overload the 12V-to-5V regulator and cause it to reset. Thus a delay mechanism was added in the form of a TPS3808 programmable delay device. Only 25ms after the 12V-to-5V regulator reaches full 5V output, do the other regulators switch on.

The PTH08 range of regulators are all switched mode regulators that provide reasonably smooth voltage conversion with low loss. As can be seen from Figure 5.7 there were also cases where linear regulators were required. The linear regulators provide relatively smoother supply voltages at the price of power loss. They are also perfect for voltage references. The MGT ports of the LX110T needed their own supply from linear regulators (1V and 1.2V supplies) due to their stringent signal integrity requirements (see Section 2 of [25]). These were catered for by the three TPS74401 devices. The QDRII+ memory modules required stable 0.75V reference voltages for their HSTL-I signalling, which were provided using two TPS51100 devices (one for each FPGA's matching bank of memory modules).

## 5.3 Final Design

The final overall design is illustrated in Figure 5.8.

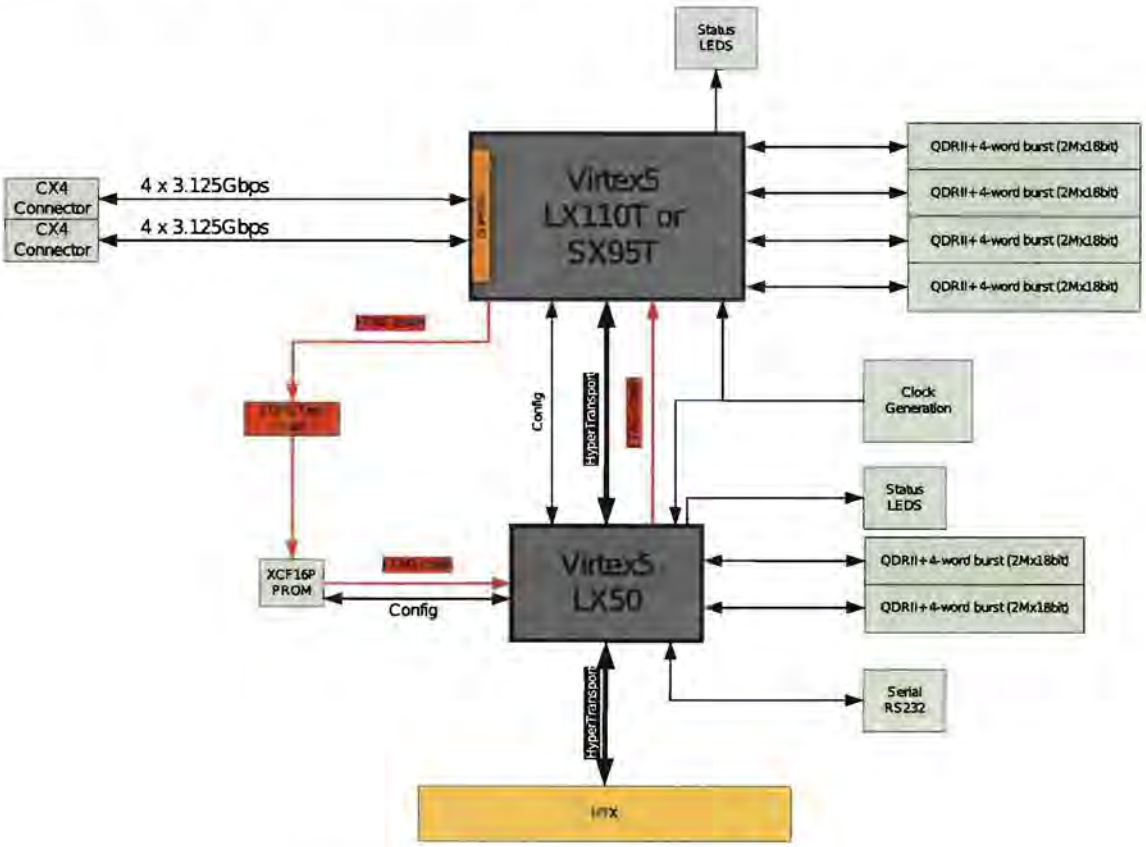



Figure 5.8: High level system overview

# Chapter 6

# Hardware Implementation

This chapter describes the process of

- capturing the design of the accelerator card
- drawing up a bill of materials
- arranging the outsourcing of the layout design
- performing simulated signal integrity tests on the layout design
- preparing the design for PCB fabrication
- arranging the component placement and assembly of the card



## 6.1 CAD Suite

The CAD Suite used to implement our design was the *Mentor Graphics PADS Solutions* software package[1]. This suite included the following programs which each played a vital role in the PCB design:

**I/O-Designer** A software tool that helps manage the large numbers of pins that are available on FPGAs.

**DxDesigner** The complete environment for capturing and creating a schematic of an electronic system.

**HyperLynx** A simulator and with associated analytical tools that help discover layout design problems such as low signal integrity and crosstalk.

[1]For more information visit `http://www.mentor.com/products/pcb/pads/`

## 6.2 FPGA pin management using I/O Designer

The two FPGAS we had to deal with had numerous pins - the LX110T has 1136, and the LX50 has 676. Although these pins appear physically on each chip as one large array, they are in fact divided up into global pins, banked pins and function specific pins [26]. Global pins are pins that service the entire device, such as internal voltage supplies (Vccint), ground pins (GND), and auxiliary power pins (Vccaux). Local pins are pins that service one of the device's I/O Banks, such as I/O supply pins (Vcco), voltage reference pins (Vref) and the I/O pins themselves. Each bank has designated clock input pins for driving interfaces implemented on those banks, whilst some banks have global clock input pins that can drive logic across the entire device. The LX110T has 18 I/O banks, whilst the LX50 has 13. Figures 6.1 and 6.2 illustrate the separate I/O banks on the two FPGAs. The division of the FPGAs' pins into I/O Banks provides the flexibility of allowing different signalling standards with different voltages to be implemented on different I/O Banks. In our design, this means the HT (2.5V LVDS) signals were implemented on separate I/O banks from the QDRII+ (1.5V HSTL-I) signals. Lastly, the function specific pins mentioned earlier are pins such as the MGT I/O and supply pins, which do not fall under the domain of any particular I/O bank.

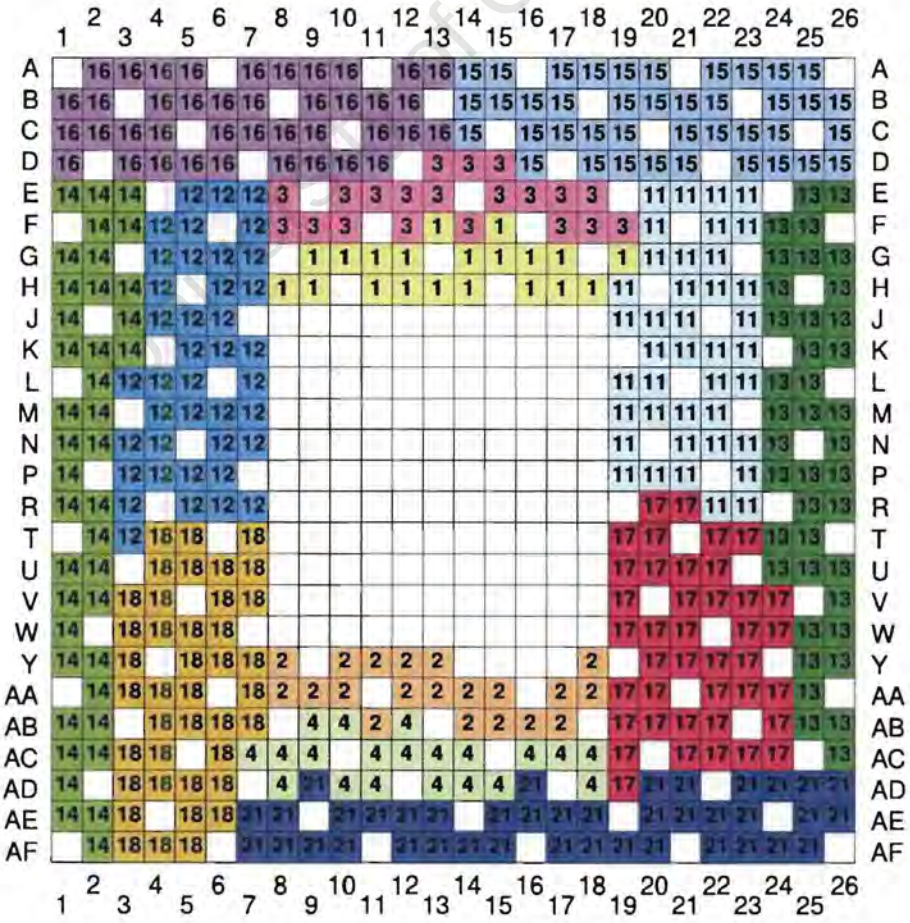


Figure 6.1: Pin-out diagram of the LX50 Virtex-5 FPGA, from [26]

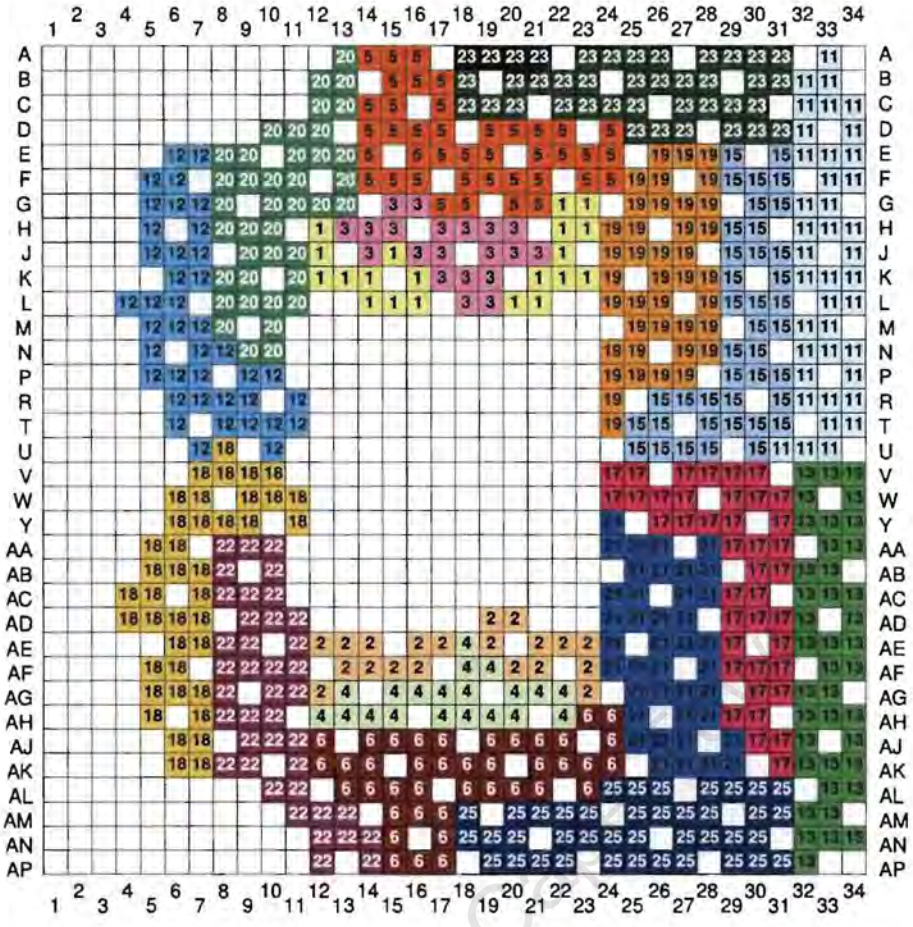


Figure 6.2: Pin-out diagram of the LX110T Virtex-5 FPGA, from [26]

I/O Designer includes a database of pin information for the Xilinx Virtex-5 range of devices. It comes with built in support for the concept of FPGA I/O banks, clock pins, global clock pins etc. This allowed us to allocate the design's different I/O interfaces to the I/O banks, select the appropriate signalling standards, simultaneously making sure that clock carrying pins were allocated to clock enabled pins. This process began by creating a list of required I/O signals and then mapping these signals, using a point-and-click interface to I/O pins on the FPGA. The end product of the process is a complete set of separate schematic symbols representing the device arranged either by I/O bank or by interface. These symbols were then imported into DxDesigner.

## 6.3 Manual symbol creation using DxDesigner

For each IC in the design (other then the FPGAs), schematic symbols had to be created from scratch to represent and encapsulate the logical representation of those devices. The created symbols also contained a list of specifications for that device such as manufacturer, distributor, package and so forth. The information needed to create these symbols was obtained directly from the component datasheets. Whilst the process of capturing the small device symbols was facilitated by DxDesigner's

Symbol Wizard, the larger devices such as the QDRII+ modules and the HTX connector had to be created using spreadsheets with lists of the device pins, pin parameters and mappings to the resulting symbol image. These tabulated data was then imported into the symbol wizard. This was to avoid tedious and error prone pin creation in the Symbol Wizard.

## 6.4 Modular design in DxDesigner

Using DxDesigner's built-in support for hierarchical designs, we first partitioned the design into modular blocks according to function. The main functional blocks that we identified were:

**HTX** The HTX connector and its decoupling capacitors and signals.

**POWER** All the switched regulators, linear regulators, power-on delay, decoupling capacitors and other devices associated with generating supply voltages and voltage references.

**CLOCKING** All the crystal oscillators and clock distributors with associated circuitry.

**JTAG** The JTAG connector and its control and data signals.

**RS232** The serial debug pins and the RS232 serial transceiver module.

**CX4_SUBSYSTEM** The CX4 connectors, decoupling capacitors and the surrounding power-sensing circuitry for sensing and powering optional optical transceivers.

**QDR_MEM_1** The group of QDRII+ memory modules numbered 0 and 1 and their termination resistors and decoupling circuitry. These are the modules which interface to the LX50 FPGA.

**QDR_MEM_2** The group of QDRII+ memory modules numbered 2 to 5 and their termination resistors and decoupling circuitry. These are the modules which interface to the LX110T FPGA.

**LX50_FF676** All the FPGA symbols, decoupling capacitors, configuration PROM and jumpers associated with the LX50 FPGA.

**LX110T_FF1136** All the FPGA symbols, decoupling capacitors, MGT circuitry and jumpers associated with the LX50 FPGA.

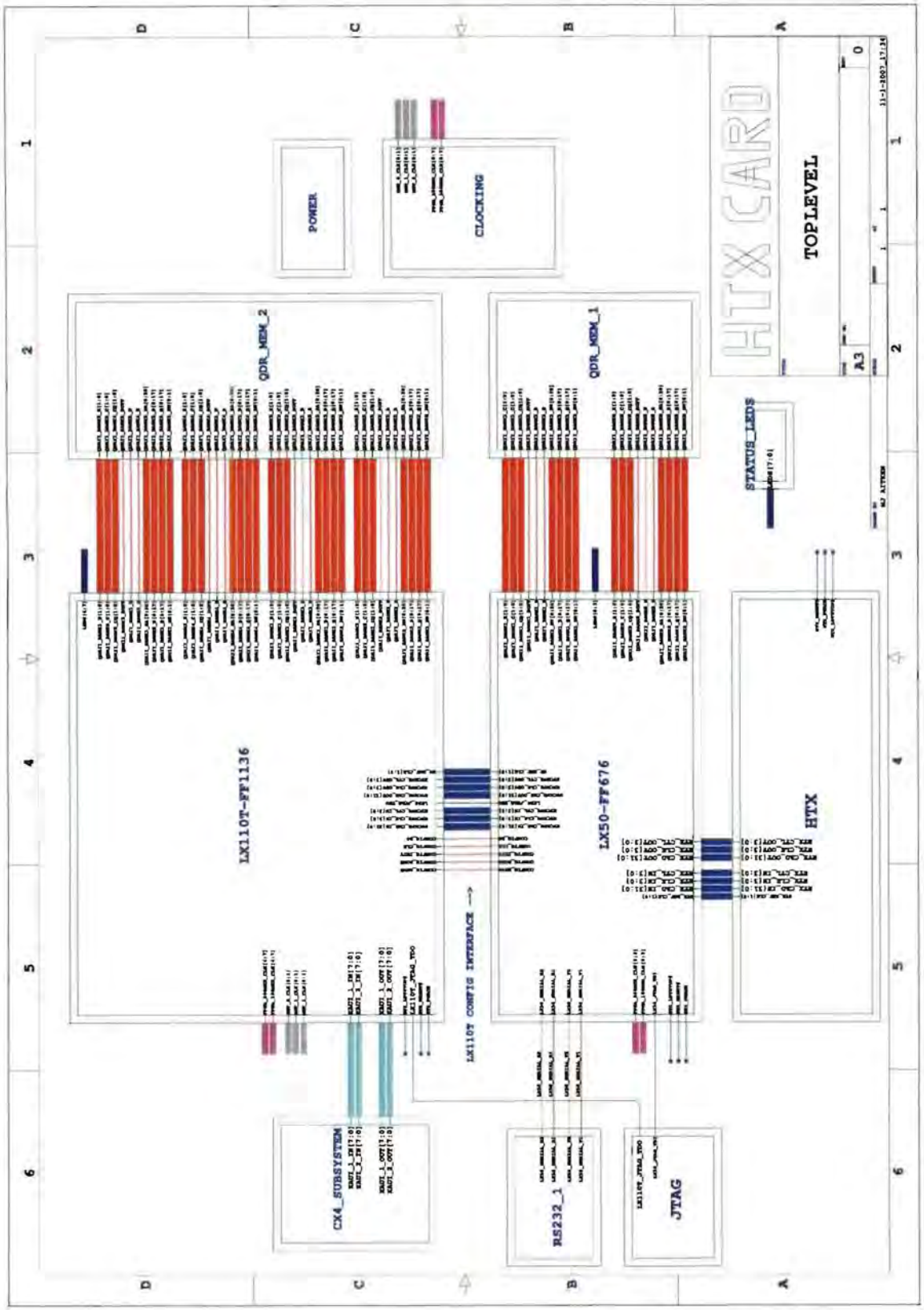



Figure 6.3: Top level schematic file showing system modules

Once all these functional blocks were identified, a top level schematic diagram was created within DxDesigner were the blocks were defined. The signals by which all the blocks interfaced to one another were then mapped out, as shown in Figure 6.3. Voltage signals that are generated within the POWER block and that carry supply current are not shown as signals on the top level schematic because they are

defined as globally available signals.

By starting the design work using a hierarchy design, we were able to abstract the low level details of the design quite easily and make changes to the top level signals between blocks with relative ease. The hierarchical blocks also allowed us to navigate the design quite easily during schematic capture.

## 6.5 Voltage Integrity

Since the system consisted of many high speed, low voltage signals in a relatively compact design, we needed to be very careful that voltage integrity was maintained by making sure that the regulators delivered smooth voltages without high frequency noise generated by the on-board components. This was accomplished by correctly calculating the required decoupling circuitry both at the input and output of the regulators, as well as at the voltage input pins of the various loads across the card. The Texas Instruments voltage regulators came with a manual which gave the calculations for determining appropriate decoupling (see [27], and the capacitance spreadsheet in Appendix A- /power_and_thermals/), whilst the Virtex-5 FPGAs came with documentation which recommends exact capacitor values, sizes and even part numbers (see [28]) for each type of supply pin. The QDRII+ memory modules did not come with decoupling suggestions and as such local decoupling for these devices was identified by mimicking a successful development board design: Xilinx's Memory Interfaces Board ML561 (see [29]). This development board also used Samsung's QDRII+ modules.

## 6.6 Outsourced Layout

The decision to outsource the layout design was taken because the design was extremely complex and dense. The scope of the project was also quite large, and it was foreseen that there would not be enough time in a two year period to do the system design, layout, and software/gateware design and implementation. Therefore while the layout work was being taken care of, focus could be given to the other components of the design.

*Mechatronics Test Equipment* is a company in Pune, India who had previously designed complex PCBs incorporating Virtex-5 devices. MTE had done work for some of the SKA/KAT's collaborators, the CASPER group at the University of California, Berkeley. They also quoted a fairly reasonable fee for the layout of approximately R60,000. Our nearest offer to this was R102,000 from a private contractor. The only drawback of using MTE, was that their CAD suite of choice was Zuken CADSTAR and as such our DxDesigner schematic design had to be translated into a different proprietary format for them to work with. Ultimately this

lead to MTE having to recapture the schematic design mostly by hand, a process which introduced some delays.

In preparation for the layout outsourcing, a comprehensive design specification was created and packaged[2]. This package contained a complete requirements specification including:

- overview diagrams
- detailed schematics
- DxDesigner CAD files
- component datasheets
- bill of materials
- track constraints
- mechanical requirements
- suggested component placement
- HTX Specification document

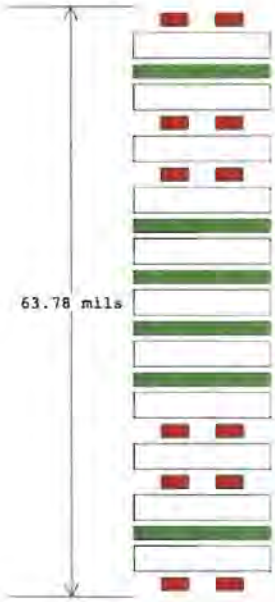
As part of their service, MTE conducted a review of our schematic design and there were no major issues that were identified. An important aspect of the layout design was meeting the signal integrity and timing constraints requirements of the high speed interfaces. For every group of high speed signals in the design, this Track Constraints document specified the following:

- signalling standard and expected operating frequency
- differential trace length matching
- intra-group length matching
- inter-group length matching
- single-ended/differential impedance
- designated clock signal/s and group-to-clock length matching

The HTX specification calls for a card with a total thickness of 1.58mm (62 mils), and this limited the number of the layers that could be implemented, a critical constraint considering the density of the design. The HyperTransport signalling standard requires paired traces with 100 Ohm differential impedance, but due to a combination of limited layer thickness, the dimensions of the FPGAs' pins, and

[2]see Appendix A - /layout_design/mte_design_specification/

the limitations of trace width for fabrication, MTE was only able to use 90 Ohm differential impedance on these traces. This compromise was considered acceptable, as designing with thinner traces to meet the 100 Ohm spec would have meant a much more expensive fabrication process that could cater for narrower traces. Figure 6.4 shows the final stack-up that was used for the design.




| NN | Layer Name | Type | Thickness mils, oz | Z0 ohm |
|---|---|---|---|---|
| 1 | Top_Elec | Metal | 1.556 | 48.8 |
| 2 | Prepreg1-2 | Dielectric | 4 | |
| 3 | GND | Metal | 1.021 | 65.2 |
| 4 | Laminate1 | Dielectric | 4 | |
| 5 | Sig_2 | Metal | 1.021 | 45.7 |
| 6 | Prepreg3-4 | Dielectric | 4.6 | |
| 7 | Sig_3 | Metal | 1.021 | 45.7 |
| 8 | Laminate2 | Dielectric | 4 | |
| 9 | +2.5V | Metal | 1.021 | 40.4 |
| 10 | Prepeg5-6 | Dielectric | 3.8 | |
| 11 | +1.5V | Metal | 1.021 | 34.4 |
| 12 | Laminate3 | Dielectric | 5 | |
| 13 | +1V_LX110T | Metal | 1.021 | 34.4 |
| 14 | Prepeg7-8 | Dielectric | 3.8 | |
| 15 | +1.8V | Metal | 1.021 | 40.4 |
| 16 | Laminate4 | Dielectric | 4 | |
| 17 | Sig_4 | Metal | 1.021 | 45.7 |
| 18 | Prepreg9-10 | Dielectric | 4.6 | |
| 19 | Sig_5 | Metal | 1.021 | 45.7 |
| 20 | Laminate5 | Dielectric | 4 | |
| 21 | GND;GND1 | Metal | 1.021 | 65.2 |
| 22 | Prepreg11-12 | Dielectric | 4 | |
| 23 | Bottom_Elec | Metal | 1.556 | 48.8 |

Figure 6.4: Diagram of stack-up

There were a few other minor changes that had to be made my MTE to complete the design. This included swapping GTP TX and RX pins to avoid creating vias (which the Virtex-5s support), and including configuration selection pins for the FPGAs. Upon completion, the layout appeared as in Figure 6.5.

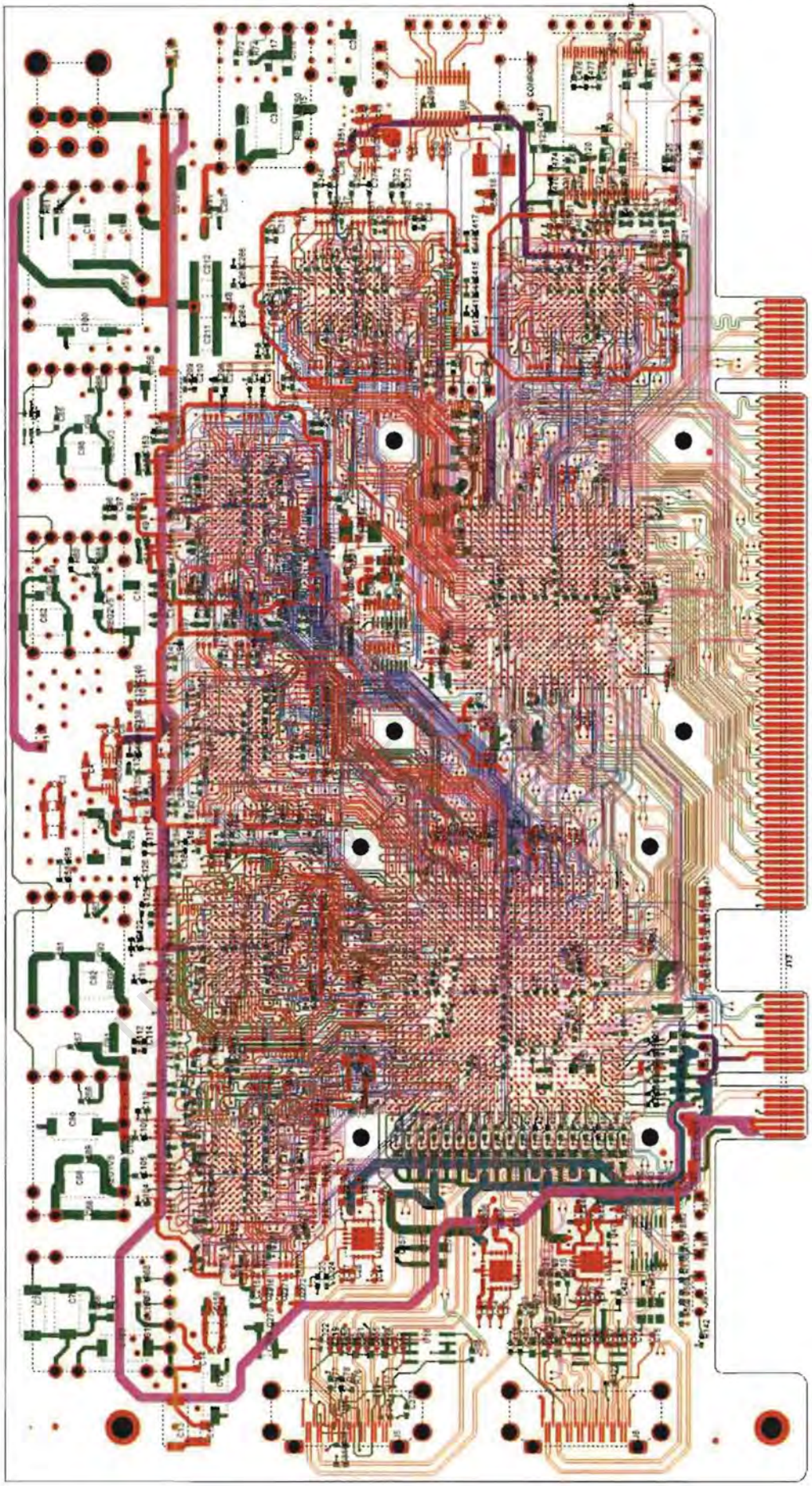


Figure 6.5: Final layout design - showing signal layers only

## 6.7 Signal Integrity Tests

Before the layout design was accepted, we had to run some simulated signal integrity tests on the design using Mentor Graphics Hyperlynx. This was done to be certain that both the timing and waveform characteristics of the high speed signals were in agreement with the track constraints handed to MTE in the design specification.

The CAD layout files (in Zuken CADSTAR format) were translated into Hyperlynx *boardsim* files using a translator. The process preserved the physical specifications of the card, including the stack-up information and the trace dimensions. However, in order to simulate waveforms on the signal traces, we needed to specify models for the transmitters and receivers for each trace. This turned out to be a laborious process that involved sifting through the boardsim files, extracting the signal names and repetitively defining models for each pin pair in a text editor. The transmitter and receiver pins could be modelled using the IBIS models downloaded from the respective IC's manufacturer website. Differential pin pairs added further complications to this process as positive and negative drivers and receivers needed to be identified.

Simulating signal waveforms is very processor-intensive and therefore time-consuming even for individual traces. Factors that effect the time for simulation include the type of simulation (standard waveform or eye-diagram), the level of accuracy required, the inclusion of crosstalk modelling, via modelling and accounting for lossy transmission. Even though there were many hundreds of high speed traces on the board, it was possible to quickly identify problem traces using HyperLynx's *Quick Analysis* function [30]. Quick Analysis allowed us to simulate the entire board in one batch simulation (with less detail per trace), the results of which then identified problem traces that required further attention. Problem traces were then modelled using a detailed simulation. The detailed simulation presented a simulated digital oscilloscope that performed many of the functions that a real oscilloscope would perform. Eye diagrams could only be created in detailed simulation and were the most time-consuming variety.

Since our design required the matching of various signals groups with their clock signals (see Section 6.6), it was necessary to verify that the groups were in fact matched to within the limits specified in the track constraints document. A spreadsheet was drawn up[3] and the results from *Quick Analysis* of the different signal groups were inserted into the spreadsheet in such a way that any timing violations would become obvious. The results of the tests are too numerous to show here. For demonstration purposes however, Table 6.1 shows the tabulation of a signal group. Notice the maximum deviation from the group CLK signal and the maximum deviation within the group of 176ps, which is within the HT specification of 350ps. Figure 6.6 shows a trace pair from the table and Figure 6.7 gives the resultant

[3]see Appendix A - /layout_design/hyperlynx_board_model/TRACK_ANALYSIS.xls

detailed simulation of the trace pair in question.

| NET NAME | Metal Delay (ps) | Impedance (Ω) |
|---|---|---|
| HTCAVE_CAD_IN01, HTCAVE_CAD_IN00 | 744.48 | 53.1 |
| HTCAVE_CAD_IN02, HTCAVE_CAD_IN03 | 741.05 | 53.1 |
| HTCAVE_CAD_IN05, HTCAVE_CAD_IN04 | 745.35 | 52.7 |
| HTCAVE_CAD_IN06, HTCAVE_CAD_IN07 | 797.23 | 52.8 |
| HTCAVE_CAD_IN09, HTCAVE_CAD_IN08 | 632.22 | 53.5 |
| HTCAVE_CAD_IN10, HTCAVE_CAD_IN11 | 620.46 | 53.5 |
| HTCAVE_CAD_IN12, HTCAVE_CAD_IN13 | 726.06 | 53.4 |
| HTCAVE_CAD_IN15, HTCAVE_CAD_IN14 | 770.89 | 53.4 |
| HTCAVE_CLK_IN01, HTCAVE_CLK_IN00 | 753.99 | 52.6 |
| HTCAVE_CTL_IN00, HTCAVE_CTL_IN01 | 729.83 | 53.4 |
| | | |
| Min | 620.46 | 52.6 |
| Max | 797.23 | 53.5 |
| Max - Min [Deviation] | 176.77 | 0.9 |
| Max Deviation from CLK | 133.54 | |

Table 6.1: Quick Analysis results of the a single HyperTransport CAD group at 800MHz.

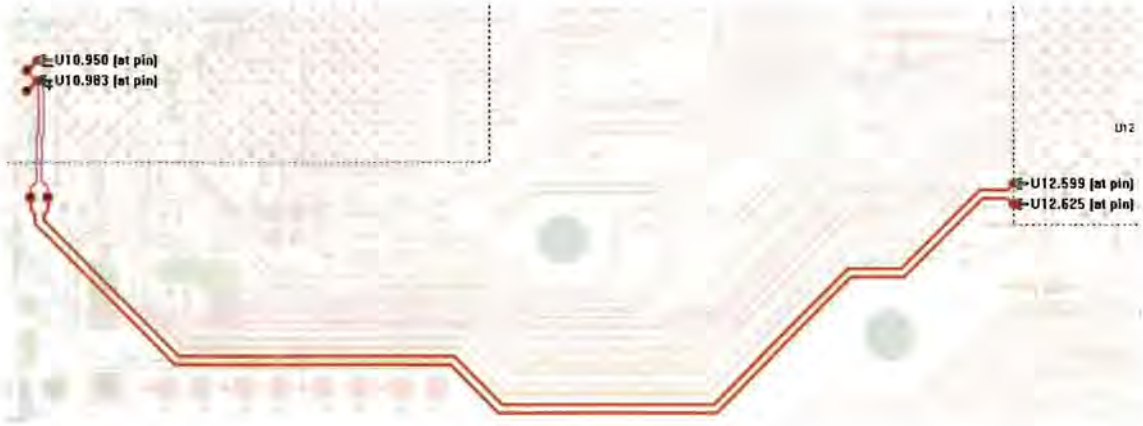



Figure 6.6: Differential traces HTCAVE_CAD_IN00, HTCAVE_CAD_IN01 shown in the pin selection screen

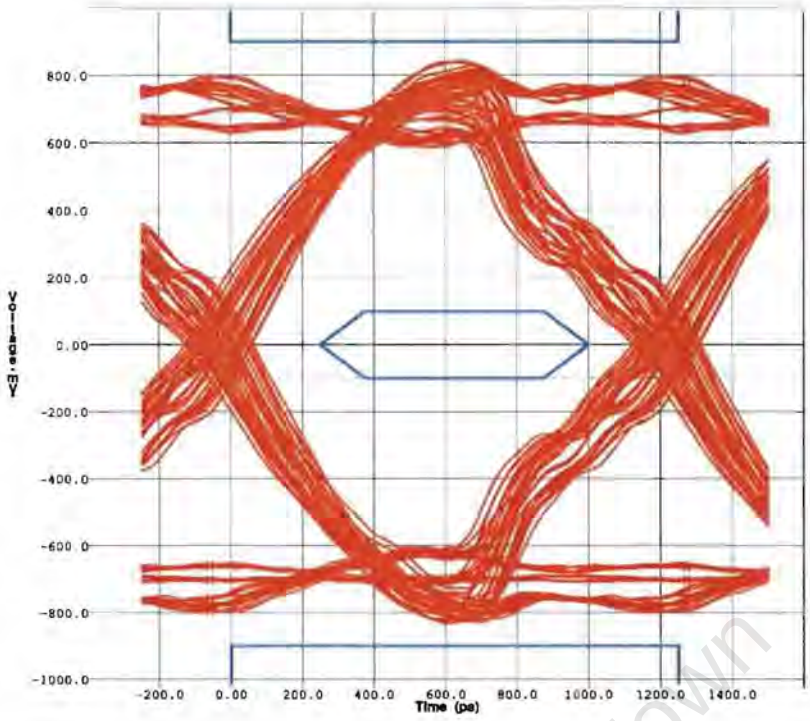



Figure 6.7: Eye-diagram of traces HTCAVE_CAD_IN00, HTCAVE_CAD_IN01 simulated at 800MHz using detailed simulation in the digital oscilloscope with 15% clock jitter

There were two main high speed signal groups that needed attention in terms of of simulation: the HyperTransport links and the numerous QDRII interfaces. To all these signals we applied the above procedure to identify problem pins. This included choosing to run the Quick Analysis using crosstalk simulations to see whether there was sufficient crosstalk between any tracks to cause degraded signal integrity. This option seemed to over-estimate crosstalk - detailed simulations with crosstalk enabled always showed that the quick crosstalk figures were completely inaccurate. Thus the crosstalk figures produced in Quick Analysis were ignored. The XAUI/10GbE links were not possible to simulate as there was no licensed software available to simulate the RocketIO transceivers that drive those traces. However, due to the low complexity of the routing for those serial links, the chances of problems arising where slim. Appendix B shows more examples of the simulation results for the HTX and QDRII interfaces as well as the clock signal traces from the crystal oscillators.

## 6.8 Bill of Materials

The various components for the board were obtained through three different channels: the FPGAs were donated, the other ICs were obtained through sampling services and the passive components were purchased from a distributor.

**FPGAs** Xilinx kindly donated four of each of the FPGAs required for the card. This meant we could assemble four prototype cards. However, we decided not

to use all eight devices at once as they were valuable, and we felt it would only be necessary to assemble two cards. This would allow us the flexibility of having reserve FPGAs, whilst still having two assembled boards to be sure that any possible failures were in fact design flaws rather then manufacturing imperfections.

**ICs** The other ICs were obtained by requesting free samples from *Texas Instruments*, *Maxim/Dallis Semiconductor* and *Linear Technology*. Where the number of samples available from these services were limited, we used *EBV Elektronik* to obtain more samples. The ICs were packaged individually.

**Passives** All the passive components were sourced from *Digi-Key*, a major electronic component distributor. Most of the passive components were packaged in cut tapes, so that they could be used with pick and place soldering machines during assembly (see Section 6.10).

Component sourcing was done mostly after the finalisation of the layout design. As such, we found that the 49 Ohm 1% tolerance resistor-arrays that had been used to terminate the QDRII+ modules had become obsolete. This required using 51 Ohm 5% tolerance resistor-arrays instead, a compromise that avoided significant redesign in the layout of the board, at the possible cost of signal integrity.

For a complete and final bill of materials see Appendix A - /assembly/HTX_BOM.ods.

## 6.9 Fabrication

We wanted to produce 5 PCBs. This would allow for two PCBs to be used for actual assembly, one for a destructive pre-assembly regulator test, and two additional PCBs for additional assemblies at a later stage. Quotations were requested from three different PCB manufacturers: StreamLineCircuits ($ 4265)[4], Hi-Q Electronics ($1831)[5] and EELab [6]. StreamLineCircuits were selected because of their experience with producing complex multi-layer PCBs with BGA pads as well as the boards they have produced for the SKA/KAT project and the CASPER group[7]. To produce the PCBs, StreamLineCircuits needed:

- The Gerber files produced during the layout design, including: power planes, ground planes, signal layers, solder & paste masks, drill details & mapping, silkscreens

[4]StreamLineCircuits of Santa Clara CA quoted for 5 PCBs plus electrical test

[5]Hi-Q Electronics of Tamilnadu, India quoted for 4 PCBs with electrical test but with limitations on the minimum trace widths/spacing.

[6]EELab, a Chinese manufacturer did not respond for quotation, presumably because they were unable to meet with our design constraints.

[7]StreamLineCircuits were responsible for producing the ROACH hardware, a PCB which was designed by SKA/KAT in collaboration with CASPER. See http://casper.berkeley.edu/wiki/index.php?title=ROACH for further reading.

- Fabrication notes which give instructions to the PCB manufacturer as to:
  - which standards the board must comply
  - which pad finishes to use
  - what colour the solder mask & silkscreens must be
  - dielectric materials
  - edge finger plating (gold)
  - finished board dimensions
- A detailed layer stack-up which describes the thickness, dielectrics and target trace impedances of each layer along with its matching Gerber file.

After fabrication, the boards gold edge fingers needed to be shaved at a 45' angle to comply with the HTX specification. This requirement was also specified in the fabrication notes.

Streamline circuits requested to change the 12 mil drill size specified in the layout design, to a 10 mil drill size. The original drill size would have caused annular ring violations in their manufacturing process. This was deemed acceptable as it would not cause any signal integrity issues.

The fabrication process took approximately 15 days to complete. The order included a solder compliance test as well as an electrical flying probe test of the various signals on the board to check that they complied with the trace impedances specified in the stack-up.

## 6.10 Assembly

Before the full assembly was carried out, a pre-assembly destructive test was done using one of the five PCBs. The seven switched regulators where soldered onto the board, along with spare decoupling capacitors and the programmable delay IC (TPS3808). The board was provided with 12V via a power supply and powered up. The voltages at various test points were measured and found to be compliant to within 1%.

A local company, Tellumat, was able to perform the assembly of the components onto the PCBs. However, a complication arose during the quotation process: the components that had been acquired were a combination of lead and lead-free (RoHS compliant) types, and as such had different soldering temperatures. Lead-free components are generally made to withstand a higher temperature compared to leaded components, as they need to comply with the high temperature lead-free soldering process. The leaded FPGAs in our design could withstand a maximum temperature of 210°C, whilst lead-free paste melts at 217-227°C [31]. A the same time, the

lead-free QDRII+ modules required a lead-free solder process as a low temperature lead solder process would not allow their BGA solder balls to melt and wet to the PCB's pad surface. Tellumat suggested a tin/lead process which would guarantee the FPGAs would solder within their maximum temperature range, whilst giving the lead-free BGAs the best chance of soldering properly. We felt this would be a good compromise.

During the passive component assembly process it was discovered that the termination resistors for the QDRII+ memory banks were too large for their pads. This was a design fault by MTE, and in order to avoid delays, the decision was made to use single 0402 resistors in their place. Fortunately, Tellumat's pick and place machine was capable of placing 0402 components very close together so we could use the original resistor array pads.

In addition another last minute change was made: the 200 MHz crystal oscillator was supplemented by a 250 MHz oscillator, due to our realisation that the 200 MHz could easily be synthesised using a digital clock manager within the FPGA, and that 250MHz would be more useful as an outside source.

After the final assembled cards were returned from Tellumat (see Figure 6.8), the following checks were carried out:

- all polarised caps orientated correctly
- all ICs orientated correctly
- check for solder bridges using a microscope
- conductivity test between separate voltage planes

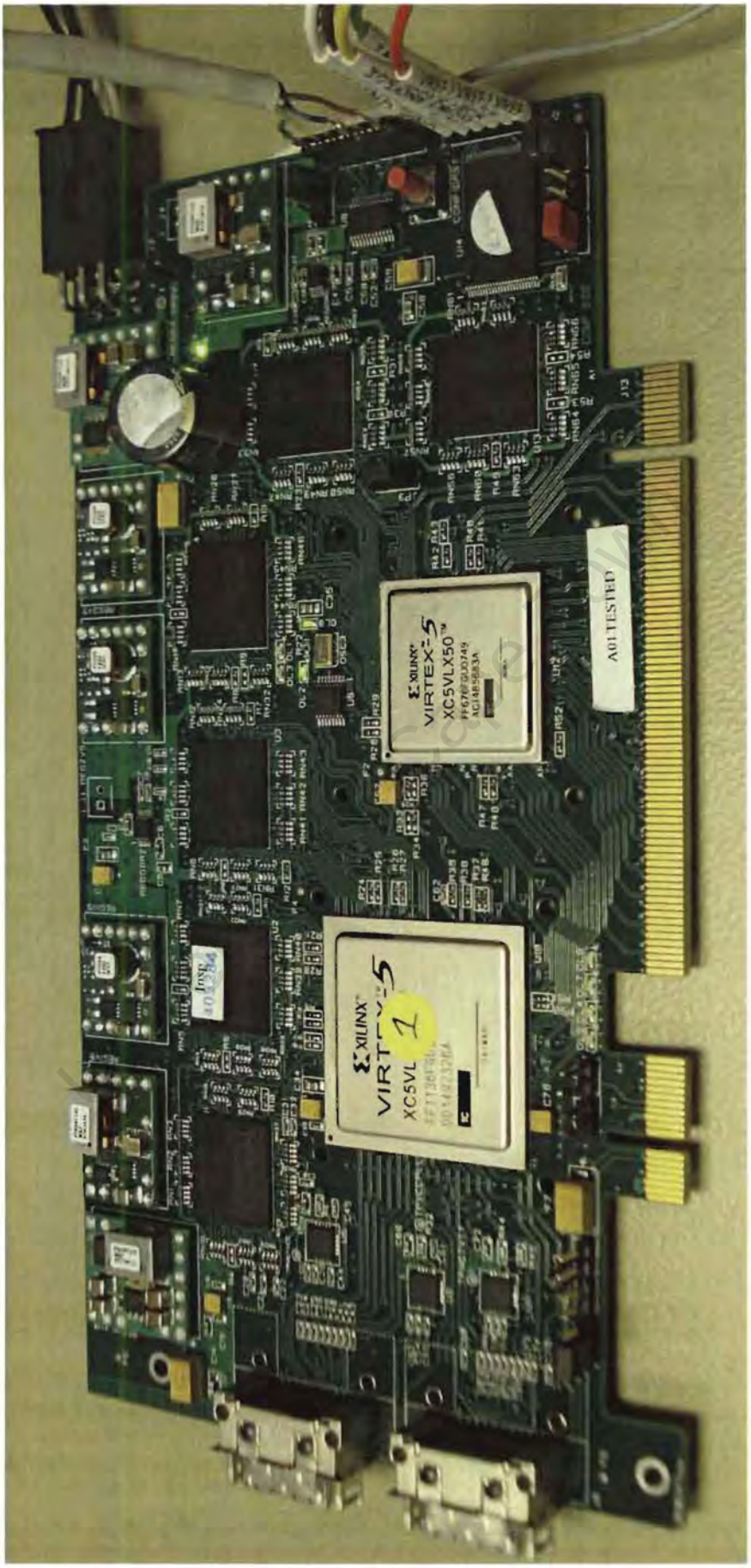



Figure 6.8: One of the two assembled PCBs

# Chapter 7

# Gateware and Interface Tests

This project involved the creation of basic gateware in order to test the accelerator card. The majority of the gateware was done whilst the PCB layout work was being carried out by MTE. In this chapter we discuss the development of the gateware to support the various I/O interfaces on the accelerator card, and then we present the results after running this gateware on the FPGAs.

## 7.1 Test Bench Setup

The card was powered using a 12V power supply through the 6-pin PCIe connector. A Xilinx DLC9G USB programming cable with flying wires and Xilinx's IMPACT software tool was used to program the bit-streams to the card for testing. Before commencing with the gateware tests, the following jumper settings were put in place:

- J6 - pin 1 and 2 connected together
- F*M* all have their 2 pins connected together
- JP3 has pins 1 and 2 connected together
- J9 has pins 2 and 3 connected together

## 7.2 Gateware

In order to map signals internal to the FPGA onto the physical pins beneath the FPGA, Xilinx requires a User Constraints File (UCF) [32] that simultaneously holds timing and location constraints for all the mappable components in a gateware design. This was the first step in the gateware design process, by mapping all the signal/pin combinations into two UCF text files (one for each FPGA) using the following format:

```
NET <SIGNAL_NAME> LOC=<PIN_NUMBER> | IOSTANDARD=<IO_STANDARD>
```

I/O Designer allows for automatic generation of a Xilinx UCF file directly from the its own signal-to-pin mappings. However, due to MTE's use of a different CAD

suite and their editing of the original pin assignments, the UCF file had to be created manually which was a tedious process considering the number of pins involved. Once the overall UCF files for each FPGA were created, these listings were then copied and pasted into individual UCF files for the specific gateware tests. Other Virtex-5 primitives such as delay calibration units and digital clock managers can also be constrained to specific parts of the FPGA using the constraints file, and where necessary this was done for the individual gateware tests. The source code and constraints files for all the gateware discussed in this chapter can be found in Appendix A.

### 7.2.1 Configuration, LED and Clock Test

In order to test that the FPGAs were placed correctly to allow configuration, simple gateware was produced that toggled the four LEDs interfaced with each FPGA. A four bit counter was created, driven by either the 100MHz and 250MHz clocks. The counter's output was applied to the LEDs so that the LEDS would change every second. Since there were four LEDS, they should reset every 16 seconds. This was a quick and easy, if inaccurate, method of deducing that the clocks were running at their correct frequencies.

The bit-streams were compiled using Xilinx's ISE, and uploaded via JTAG to the FPGAs using the IMPACT tool. Both the LX50 and the LX110T demonstrated flashing LEDS correctly, with either 100MHz or 250 MHz clocks taking exactly 16 seconds to cycle as specified. This indicated correct clock operation.

The same bit-stream for the LX50 was then repackaged using Xilinx's PROM Formatter tool for the on-board PROM device. Using JTAG, the bit-stream was downloaded and stored in the PROM device. After this, a simple power cycle was needed to test whether the LX50 could configure itself from the PROM. This process was successful, proving the PROM worked according to the design specification.

A copy of the necessary HDL files and bit-streams can be found in Appendix A - /gateware/LEDS_and_Clock_Test/.

### 7.2.2 Serial UART Interface Test

Of all the I/O controllers, the serial UART core was the easiest to prepare owing to its simplicity, popularity and availability through open-source HDL websites. A simple Verilog UART core design was obtained from a project by Jeung Joon Lee titled *Micro-UART* [33]. Lee provided good documentation with a neatly modularised core. This made the process of using the core fairly straightforward. Before the arrival of the fabricated and assembled card, the core was tested on a Xilinx Ml506 development board connected from its serial port to a host machine's serial port, via a standard cable. The core consists of the following Verilog modules (list is courtesy of [33]):

`uart.v` The top-level hierarchy module in which all the sub-modules are instantiated.

`u_xmit.v` The asynchronous transmitter, consisting of a state-machine, serialiser and support logic.

`u_rec.v` The asynchronous receiver, consisting of a dual-rank synchroniser, state-machine, deserialiser and support logic.

`u_baud.v` This is the baud-rate generator. An internal "baudclock" which is 16 times the desired baud-rate is generated from the applied external clock.

`uart_inc.h` A configuration file consisting of attributes which set the baud-rate, external clock rate and the size of the data-byte.

The micro-uart core can be driven by any system clock, and deliver any resultant baud rate by setting the following parameters in the `uart_inc.h` file before compiling the gateware:

```
parameter XTAL_CLK = <Input clock frequency in Hertz>;
parameter BAUD = <Desired board rate in bits per second>;
parameter CLK_DIV = XTAL_CLK / (BAUD * 16 * 2);
parameter CW = <cw>; // where cw >= log2(CLK_DIV)
```

For our design this is 100MHz or 250MHz depending on which input clock is used (see Section 5.1.8). The core delivers a 16 × *baud* rated clock which can be used to drive the logic that uses the core. The baud rate specified in the uart_inc.h must be matched by the baud rate specified in the serial terminal client running on the host test machine, in our case this was an open-source application called *Minicom*.

In order to test the functionality of this core, a simple loopback test was created, where any character arriving on the receiver line was transmitted back on the transmission line. Any character entered on the terminal client was echoed back, to proof the interface worked correctly. The serial port was found to be working correctly at 115200bps - the maximum standard UART bitrate.

The serial core became extremely useful in testing the remaining cores and interfaces, and we created some simple gateware that could read out register values depending on which character was pressed on the terminal client side. This helped debug the other cores.

The UART core, loopback test, and serial debug module can all be found in Appendix A - /gateware/Serial_UART_Test/.

### 7.2.3 FPGA-to-FPGA HT-Link

Gateware was required to test the connectivity between the LX50 and LX110T FPGA, referred to in Section 5.1.3. In order to do this, a simple parallel interface was implemented over the HT signals connecting the devices, with a signal generator and

comparator state machine on the LX50 and a loopback mechanism on the LX110T. This is illustrated in Figure 7.1. This interface was tested at 100MHz, just to verify the connectivity of the link. To run the link at a faster clock rate, delay matching using delay matching primitives within the Virtex-5s would be required. A HT Controller core would be expected to implement this delay matching and provide a calibration mechanism (see Section 7.2.6).

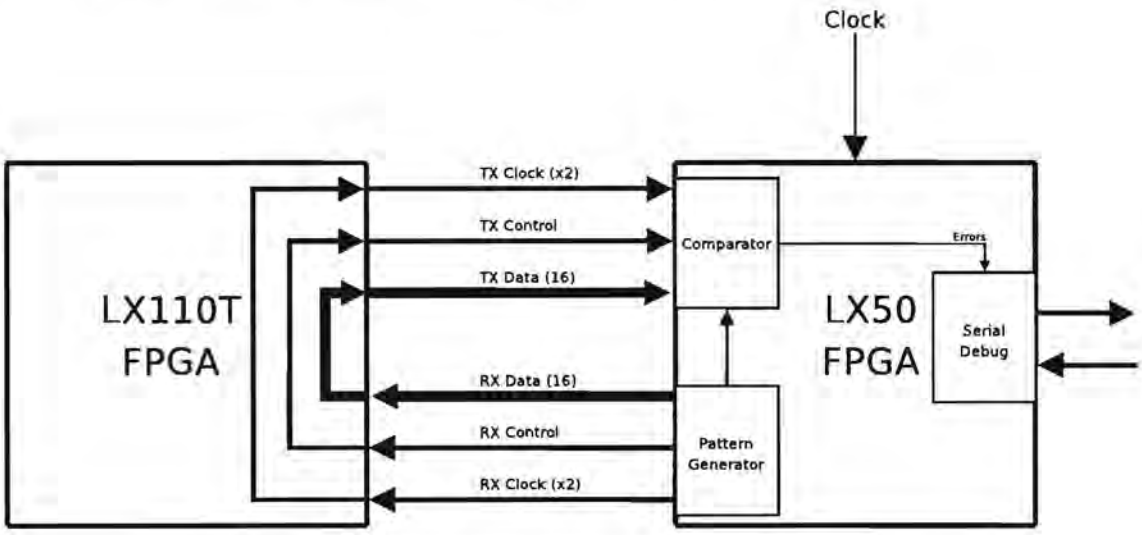



Figure 7.1: FPGA-to-FPGA HT-link test gateware

The source code and constraints for this gateware can be found in Appendix A - /gateware/HT_Link_Test/.

### 7.2.4 QDRII+ Interface

Xilinx offer a wizard for generating memory controllers called the Memory Interface Generator (MIG). We were able to use MIG 2.0 to generate a QDRII controller core with the following attributes:

- 18-bit Data bus
- 19-bit Address bus
- 250MHz capable for Virtex-5 speed-grade -1

Since our memory modules were QDRII+ devices, and the MIG was generating QDRII controllers, it was necessary to make minor changes to the HDL of the core. Fortunately Xilinx's QDRII controller did not require a license and its source was available to edit. The controller comes with a built-in mechanism which performs delay calibration on the data and control lines relative to the clocks, as well as a user test bench which continually writes test patterns to the memory module and then reads them back to detect errors. Figure 7.2 demonstrates how the core was connected up to other components. It required creating some peripheral gateware

such as a Digital Clock Manager (DCM) to create the required clock signals, buffers (not shown) and drivers for the indicator LEDS. To demonstrate that the controller and memory banks were working, the Calibration and Error signals were monitored using the LEDS.

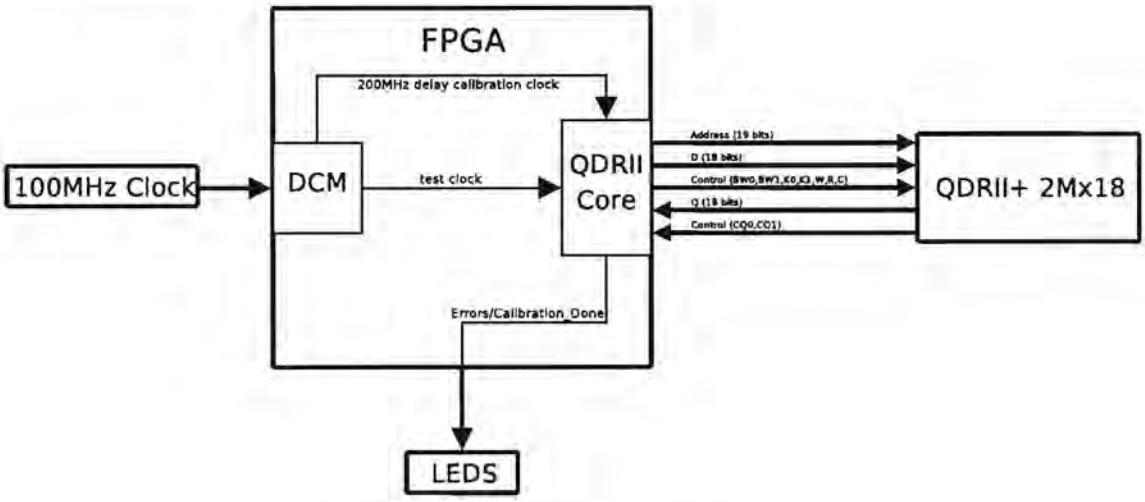



Figure 7.2: QDRII test gateware

In our tests, we found that memory banks 0,1,2,4 & 5 calibrated correctly at 150MHz, whilst memory bank 3 could only calibrate at 100MHz on both assembled cards. Since both cards had the same problem, this issue is probably due to poor signal integrity over the tracks to that bank, and an issue that was not observed in the HyperLynx simulations.

The source code and constraints for this gateware can be found in Appendix A - /gateware/QDRII_Test/.

### 7.2.5 CX4 Interface Test

In this test we set out to prove that the CX4 interface (traces and connectors) were operational by passing XAUI signals over them. XAUI is the standard for connecting a 10GbE PHY to the MAC component on a circuit board. In the case of 10GBASE-CX4, XAUI is the sublayer for the 10GbE transmission system, and thus it is the highest layer required to test the physical functionality of the 10GBASE-CX4 interface. Using Xilinx's Core Generator we implemented a XAUI v7.3 core with the following settings:

- Internal XGMII interface
- No Simplex split
- No 802.3 State machines
- No MIDO Interface
- No Elastic Buffer

Once we had implemented the constraints for cores for both CX4 interfaces, we provided peripheral components such as DCMs to deliver clock signals and a serial debug mechanism to read out the status registers of the cores. To test the interfaces we carried out two separate procedures:

1. A differential probe test was carried out on the transmission lines originating on the LX110T. The 10GSample/s probe housed at the SKA/KAT offices offered a XAUI test setup that presented the appropriate eye diagram mask for XAUI signals. Figure 7.3 shows how a CX4 cable was plugged into the connector on the card and then a spare CX4 connector with the soldered terminated probe was plugged onto the other end of the cable. The measurements taken by the oscilloscope are shown in Figure 7.4 for the 0.5m cable and Figure 7.5 for the 5m cable. The 0.5m cable test clearly shows an open eye diagram and the oscilloscope was able to lock onto the embedded clock in the signal quickly (due to XAUI's 8b/10b encoding). However, the eye diagram for the 5m cable was not open, and required tweaking of the RocketIO preemphasis and differential swing values in order to open it as much as possible. The oscilloscope was still able to lock onto the clock within the signal however it took a significantly longer time to do so. The results for the 5m cable indicated that the interface would operate over 5m cable spans but would have a substantially high error rate during normal use.

2. The same CX4 connector was looped back and plugged into the second CX4 connector on the card, as in Figure 7.6. The two XAUI cores could now talk to one another, and a sample frame was continuously transmitted from each core containing idle bits. The status bits of the two cores were read out using the card's serial port. Both were found to have healthy status bits: the receivers were synchronised to the incoming data streams for both cores within a fraction of a second from power-up.

The source code and constraints files to implement the above tests can be found in Appendix A - /gateware/XAUI_Test/.

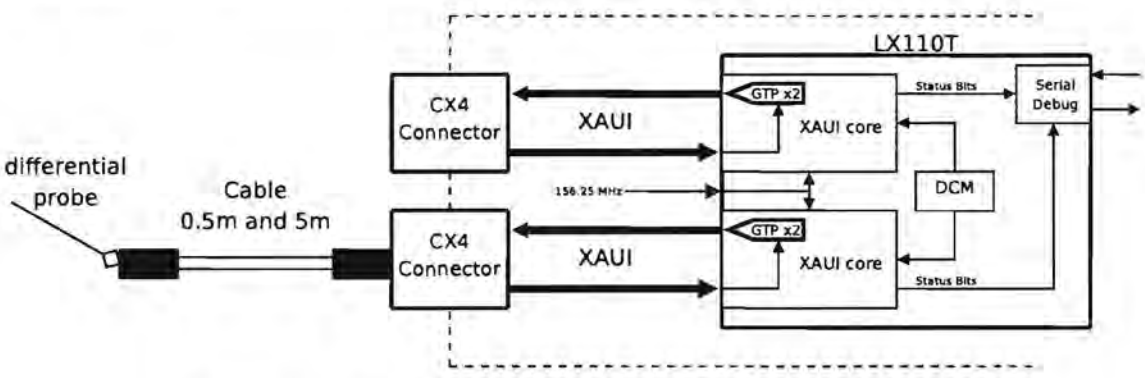



Figure 7.3: Differential probe test on CX4 XAUI signals

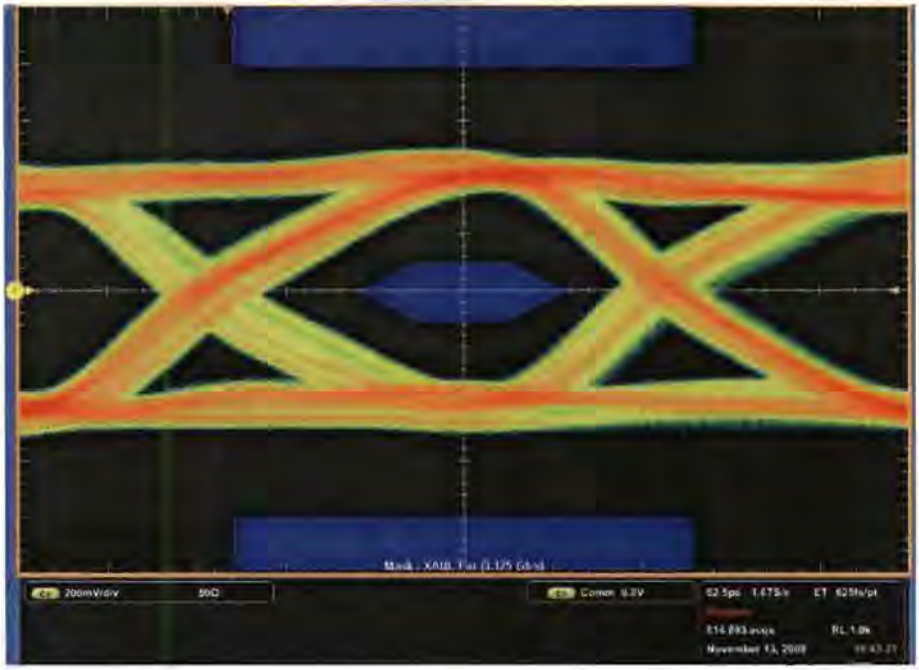


Figure 7.4: XAUI TX eye diagram probed over 0.5m cable.

### 7.2.6 HyperTransport

The HyperTransport interface is the most vital interface in the design. Since the process of converting the University of Mannheim's Virtex-4 HyperTransport controller core to our platform was a large undertaking, involving complex HDL programming and software driver development, it was deemed necessary to make this a masters project in itself. Nicholas Thorne, also of University of Cape Town under the Advanced Computer Architectures research group, was tasked with this undertaking. For further details on this subject, the reader is advised to refer to Nicholas Thorne's work on the HyperTransport core once completed, which will be published both through UCT and the ACE group.

## 7.3 Test Summary

| Test | Pass | Significant Measurements |
|---|---|---|
| FPGA Configuration | Yes | all LEDS, and clocks verified to be functional. |
| Serial UART Loopback | Yes | TX and RX via serial port at 115200bps |
| FPGA-to-FPGA HT-Link | Yes | functional at 100MHz without delay matching calibration |
| QDRII+ | Yes | 150 MHz operation on banks 0,1,2,4 & 5 only 100MHz operation on bank 3. |
| CX4 XAUI Loopback | Yes | functional over 0.5m cable functional but noisy over 5m cable. |

Table 7.1: Results of interface tests

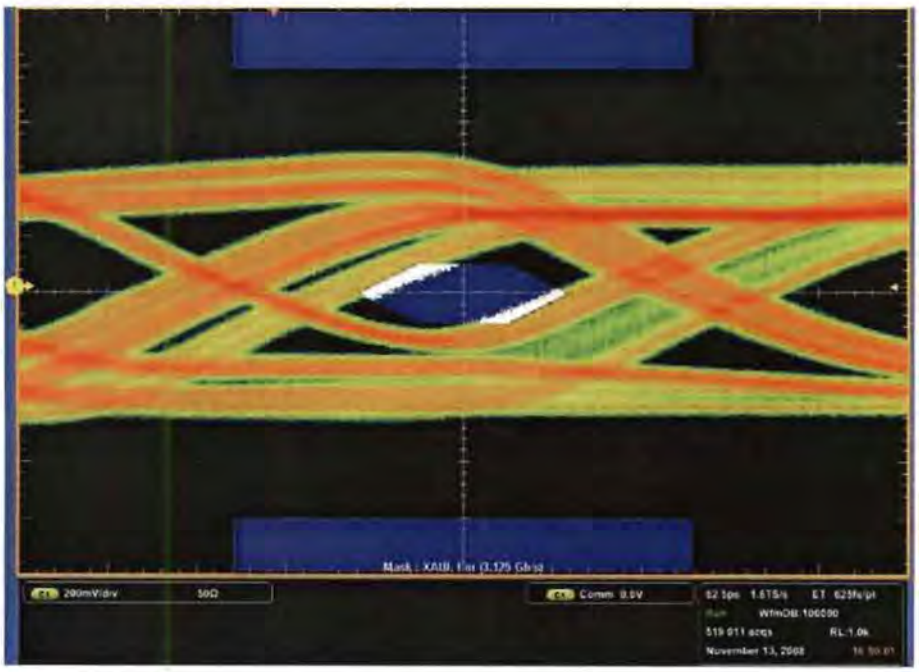


Figure 7.5: XAUI TX eye diagram probed over 5m cable.

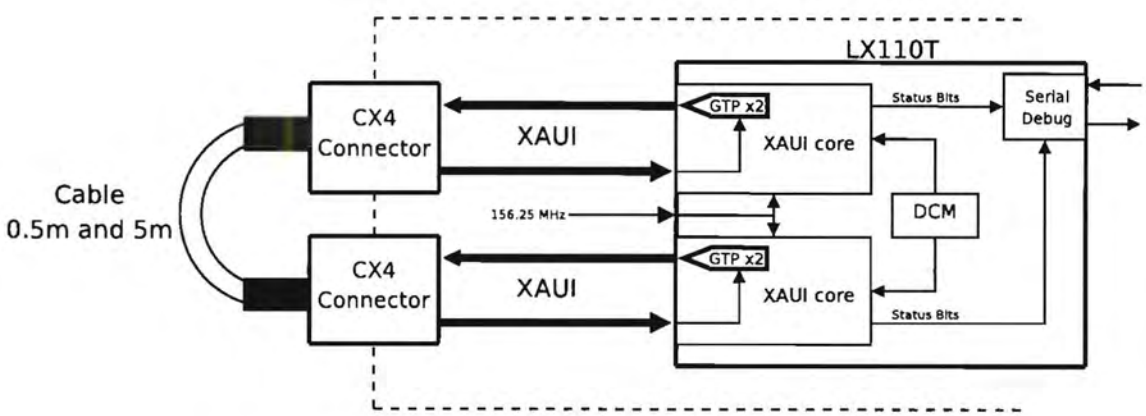



Figure 7.6: Loopback test on CX4 XAUI Interfaces

# Chapter 8

# Future Work and Recommendations

## 8.1 Further Work

### 8.1.1 Gateware Library

It would be convenient for a set of gateware components to be defined in a library for future developers to use. These would include the various controller cores used to implement the card's interfaces, as well as serial debug mechanisms that have been used in this project. Most of this gateware is already prepared, but will need to be packaged and documented correctly to make future development faster and more efficient. One important piece of missing gateware is a configuration controller on the LX50 that will allow the LX110T to be reconfigured during run-time, as per the design's intended purpose.



### 8.1.2 Drivers and API Tool Set

To make procedure calls to the card from a host machine, an API tool set along with drivers are required. This is envisaged to form part of Nicholas Thorne's work in preparing the card to be usable in a HTX-enabled host machine. The API should include:

- Routines for transmitting and receiving blocks/streams of data to/from the card via either a 10GbE network or over the HTX interface.
- Routines for reconfiguring the LX110T during run-time with new bit-streams
- Simple demonstration applications for the card
- Documentation for the above resources

### 8.1.3 Reconfigurable Accelerator Card Revision 2

There are a number of improvements to the design that could be made if a second revision of the card was produced:

- Add a flash chip to the design to store soft-core processor code (e.g. Xilinx's *MicroBlaze*)
- Improve the signal integrity of the QDRII memory interfaces to allow greater clock rates (250MHz)
- Swap the HTX interface for a PCIe 2.0 interface (would require LX50T or larger FPGA) to allow greater versatility across systems.
- Allow the LX50 to be programmed using parallel mode by the PROM (allows faster power-ups)
- Include a bracketed JTAG header to avoid using flying wires.

## 8.2 Recommended Project Extensions

Beyond getting the card working correctly on a HyperTransport bus, it would be desirable to build a reconfigurable cluster using an array of the accelerator cards. This could be done as envisaged in the use case diagrams (see Figures 4.2 & 4.3)

# Appendix A

# Design Files

Please find the DVD attached to this dissertation. Documentation referred to in the chapters can be found on the DVD.

# Appendix B

# Detailed HyperLynx Simulations

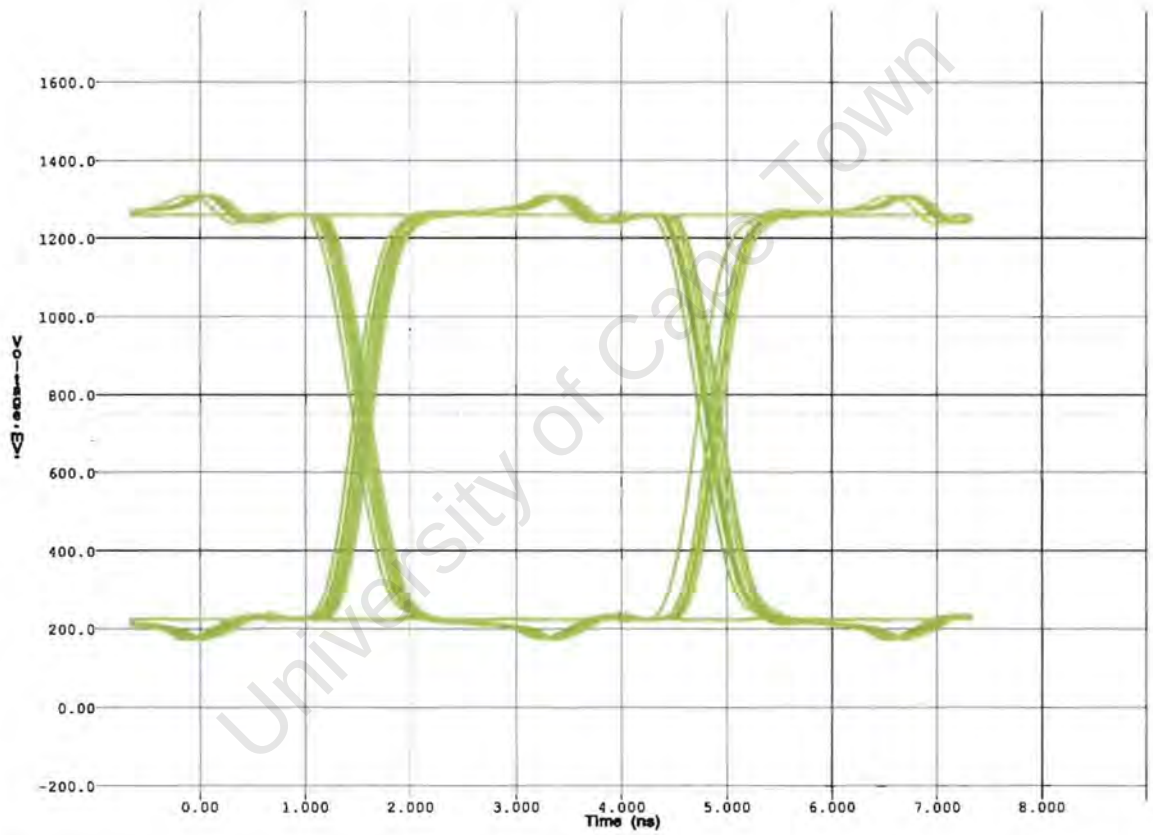



Figure B.1: Eye diagram of QDRII+ trace simulated at 300MT/s, 15% clock jitter

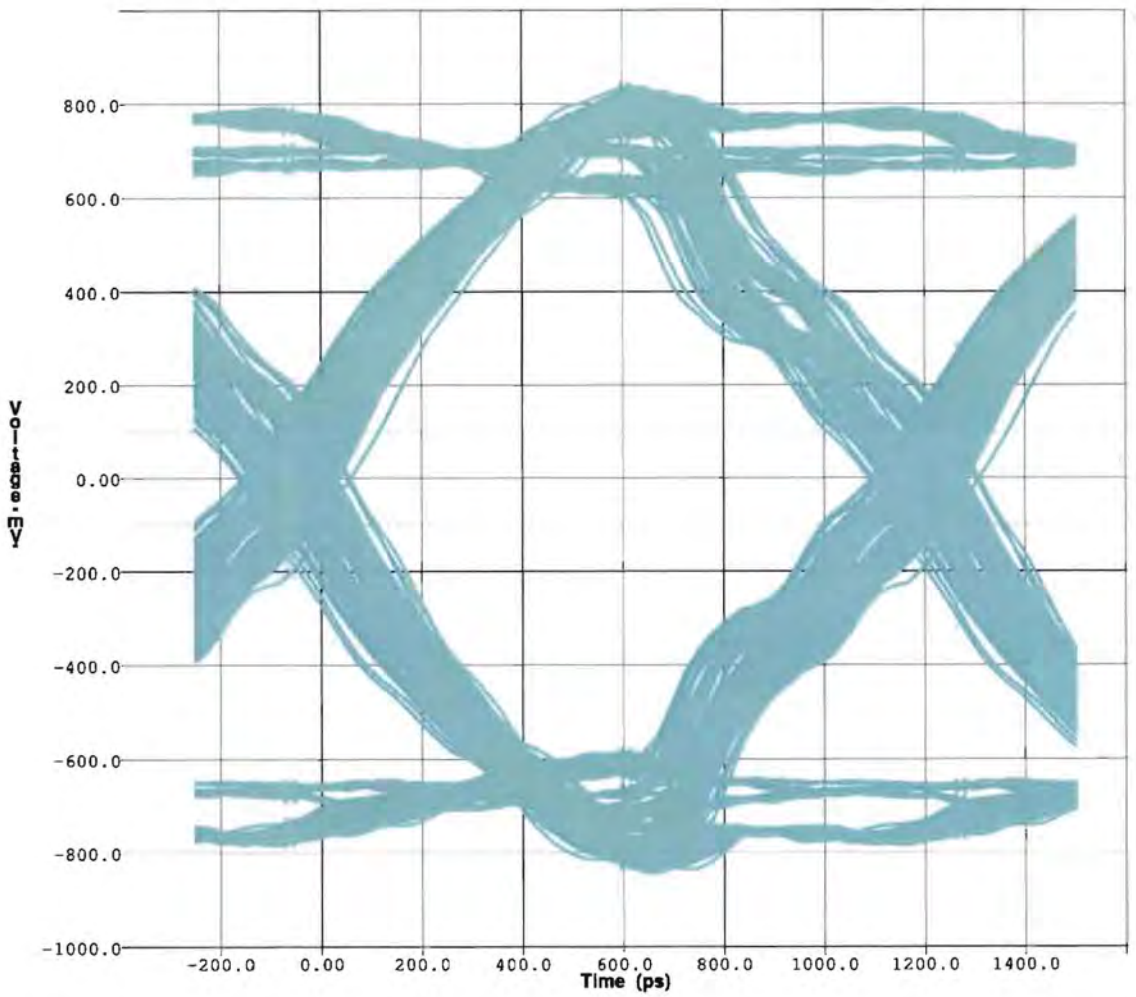



Figure B.2: Eye diagram of HTX trace simulated at 800MT/s, 15% clock jitter

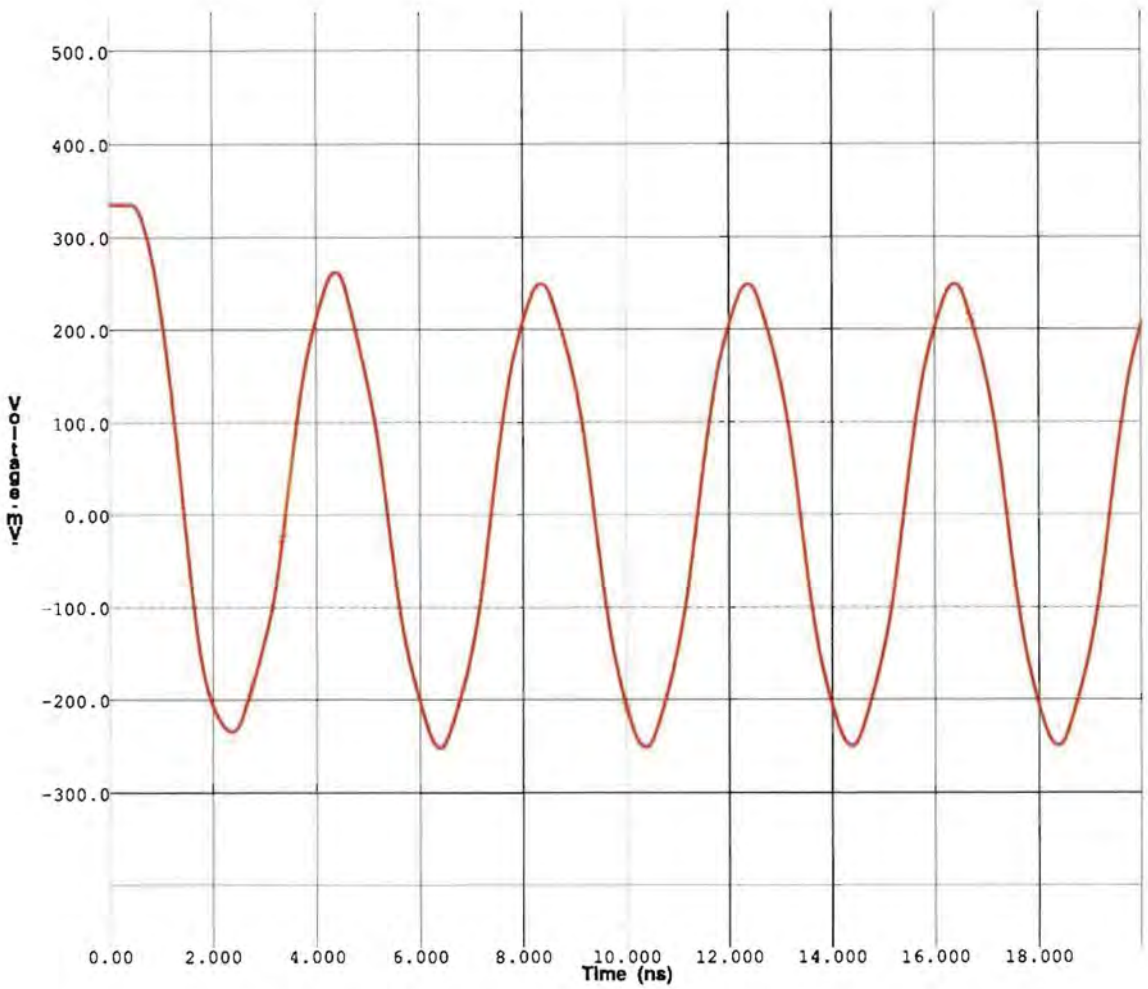



Figure B.3: 200 MHz LVDS clock signal simulation

# Bibliography


[1] H. Fröning, M. Nüssle, D. Slogsnat, H. Litz, and U. Brüning, "The HTX-Board: A Rapid Prototyping Station,"

[2] "The Open Graphics Project." `http://wiki.opengraphics.org`.

[3] G. Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, 1965.

[4] C. J. Bashe, L. R. Johnson, J. H. Palmer, and E. W. Pugh, *IBM's Early Computers*. MIT Press, 1986.

[5] J. Kilby, "Miniaturized electronic circuits," 1964.

[6] R. C. Dorf, *The Electrical Engineering Handbook*. CRC Press, 2006.

[7] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro*, pp. 23–29, July 1999.

[8] A. Wolfes, "Intel clears up post-tejas confusion," *VAR Business*, May 2004.

[9] R. Ramanathan, "Intel Multi-Core Processors: Leading the Next Digital Revolution," tech. rep., Intel Corporation, September 2005.

[10] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," pp. 79–81, 2000.

[11] T. Todman, G. Constantinides, S. Wilton, O. Mencer, W. Luk, and P. Cheung, "Reconfigurable Computing: Architectures and Design Methods," *IEE Proceedings: Computers and Digital Techniques*, vol. 152, pp. 193–207, March 2005.

[12] "Xilinx 90nm Process Technology Drives Down Costs." `http://www.xilinx.com/partners/90nm/`.

[13] B. Holden, "Latency Comparison Between HyperTransport and PCI-Express In Communications Systems," *HyperTransport Consortium whitepaper*, 2006.

[14] H. H. Ng and D. Isaacs, "Closely Coupled Co-processors for Algorithmic Acceleration," *FPGA Journal*, August 2005.

[15] K. Asanovic *et al.*, "The Landscape of Parallel Computing Research: A View from Berkeley," tech. rep., University of California Berkeley, December 2006.

[16] "Celoxica's RCHTX-XV4 High Performance Processing Card." `http://hypertransport.org/docs/tech/rchtx_datasheet_screen.pdf`.

[17] "Nallatech's BenONE-PCIe FPGA computing flatform." `http://www.nallatech.com`.

[18] "Pico Computing's E-16 LX50 ExpressCard." `http://www.picocomputing.com`.

[19] "HyperTransport I/O Link Specification - Rev 3.00." `http://www.hypertransport.org`, June 2007.

[20] "HTX Connector and Form Factor Specification for HyperTransport Daughtercards and ATX/EATX Motherboards." `http://www.hypertransport.org`, April 2007.

[21] "PCI Express Base Specification Revision 1.1." `http://www.pcisig.com/`, March 2005.

[22] *IEEE Std 802.3ae Standard.*

[23] Xilinx, Inc., *DS100 - Virtex-5 Family Overview*, June 2008. v4.3.

[24] Xilinx, Inc., *UG150 - LogiCORE IP XAUI v7.3*, March 2008.

[25] Xilinx, Inc., *UG196 - Virtex-5 RocketIO GTP Transceiver User Guide*, May 2007. v1.3.

[26] Xilinx, Inc., *UG195 - Virtex-5 Packaging and Pinout Specification*, March 2007. v3.1.

[27] Texas Instruments, *SLTA055 - Input and Output Capacitor Selection*, February 2006.

[28] Xilinx, Inc., *UG203 - Virtex-5 PCB Designer's Guide*, December 2006. v1.0.

[29] Xilinx, Inc., *ML561 : LXT-based Reference Design*, January 2007. Revision C1.

[30] Mentor Graphics Corporation, *BoardSim User Guide*, October 2006. Revision 1.1.

[31] Xilinx, Inc., *UG112 - Device Package User Guide*, May 2007. v3.0.

[32] Xilinx, Inc., *Xilinx ISE 9.2i Software Manuals - Constraints Guide*, 2007.

[33] "Micro-UART." http://www.cmosexod.com/ip/u-uart/micro-uart.PDF.

[34] J. Rabaey, *Digital Integrated Circuits.* Prentice Hall, 1996.

[35] G. Stitt, F. Vahid, and S. Nematbakhsh, "Energy savings and speedups from partitioning critical software loops to hardware in embedded systems," *ACM Trans. on Embedded Computing Systems*, vol. 3, pp. 218–232, February 2004.