Firmware and Gateware for the ACE1 Reconfigurable Accelerator Card

Nicholas James Thorne

A dissertation submitted to the Department of Electrical Engineering,
University of Cape Town, in fulfilment of the requirements
for the degree of Master of Science in Engineering.

Cape Town, 2011

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in Engineering in 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

10 January 2011

Abstract

This thesis describes the continued work on the in-house designed FPGA based co-processor daughtercard referred to as ACE1. The design of the board was done by Michael Aitken and resulted in a hardware platform with limited testing completed and limited functionality available to users. The original aim of this project was to create an ecosystem incorporating firmware, bootstrapping code, drivers and a development environment. The intention was to create a seamless environment to allow scientists to utilize the ACE1 resources in a simple, consistent and efficient manner by pulling together a variety of software and hardware from a number of sources. An overview of the board is the starting point, with a review of some of the requirements that contributed to the final design. Following the overview are the efforts that went into setting up and subsequently debugging the interface that connects the co-processor daughtercard to the host server. The challenges faced include: problems with the power network, the edge connectors and finally, timing problems with the primary protocol which prevented host-based communications.

From this point the project diverges and the options include allowing the daughtercard to function in a stand-alone fashion and we present a gateware solution for assisting users in utilizing the networking interfaces found on the daughtercard. We present our gateware options such that users can select from a number of alternatives available for each of the layers in the Open Systems Interconnect networking model. As time constraints did not allow for a complete conversion to a stand-alone processing board, we outline the future steps required to make the most of the sizeable investment in ACE1.

The alternative to utilizing ACE1 as a stand-alone processing board is to re-work the daughtercard, taking into account some of the new products and tools that have arrived in the FPGA co-processing market place since the design of ACE1 was finalized. We provide reviews of these products which clearly show the direction that the industry has taken in designing host-based co-processor systems. Some of the re-work issues were already understood at the time that Michael Aitken concluded his contribution to the ACE1 project and were presented as future work. When looking at the option of creating a second version of the co-processor board we provide a few newer ideas, sparked by the advances in the partial reconfiguration toolchains, that we discuss and recommend.

To the people in my life

who always

believe in me

Acknowledgements

I am indebted to a great number of people who through their contributions, encouragement and support have made this thesis possible.

Firstly to the Council for Scientific and Industrial Research and more specifically the Meraka Operating Unit whose Human Capital Development program, initiated by Professor Mike Inggs, offered me the opportunity and resources to tackle the challenge of a Master of Science in Electrical Engineering. I am extremely grateful to them for the support and funding of my project.

To the employees based at the CHPC who made working there a real pleasure and made the ACE research group possible:
Dr Happy Sithole, Lesley Fredericks, Chris Petzer, Kevin Colville, Sebastian Wyngaard and Eric Mbele. A special thanks to Dr Jeff Chen for always supporting and believing in the work done by the ACE research group, for always being a great role model and leading us by example, and lastly for always having time for our problems.

Further thanks to Professor Mike Inggs who got me involved in research at the CHPC and for being my supervisor for this project.

Also greatly appreciated were the efforts by Alexander Giese from the University of Heidelberg - Computer Architecture Group for your assistance in debugging the problems with the host-based interface on ACE1.

To the many students who have come and gone from ACE lab during my time there - I have had a lot of fun working alongside all of you. I've enjoyed the arguments, the games, the challenges, the companionship and the friendship that you have offered me during my time in the ACE lab.

Finally to Shannon Filmer who has supported me all through the project including and especially during the times that I struggled the most. Your support and assistance has been invaluable to me.

Contents

List of Figures

List of Tables

Nomenclature

10GbE IEEE 10 Gigabit Ethernet

ACE Advanced Computer Engineering

ACE1 Advanced Computer Engineering Laboratory's reconfigurable accelerator card revision 1

AES Advanced Encryption Standard

BGA Ball Grid Array

BRAM Block RAM - fixed sized, dual ported, RAM blocks located inside the FPGA

CCLK Configuration CLocK

CHPC Center for High Performance Computing

DMA Direct Memory Access

DRAM Dynamic Random Access Memory

FET Field-Effect Transistor

FLOPS FLoating point Operations Per Second

FPGA Field-programmable gate array

FSB Front Side Bus

GPGPU General Purpose Graphics Processing Unit is the term used when we use the GPU pipeline to solve mathematical calculations rather than render pixels

GPU Graphics Processing Unit

HPC High Performance Computing

ILA Integrated Logic Analyzers - Xilinx provide a debugging environment allowing Integrated Logic Analyzers to be attached to hardware for debugging

IP Intellectual property

MAC Media Access Controller

MIG Memory Interface Generator

NDA Non-Disclosure Agreement

NRE cost Non-Recurring Engineering cost is the cost associated with the engineers time required to create a product. It usually consists of everything other than raw-materials and production line orientated costs. It has to be re-couped by dividing it by the number of units and adding it to the unit price.

PHY is an abbreviation for the physical layer of the OSI model.

QDR Quad Data Rate

SERDES SERialize/DE-Serialize hardware blocks found at the edges of Virtex 5 FPGAs allowing higher speed IOs than would be available without fixed hardware.

soft-IP CPU A CPU architecture synthesised using an FPGA tool-chain synthesiser and implemented in FPGA fabric resources.

SSD Solid State Disk

SSL Secure Socket Layer

TCO Total Cost of Ownership

TLS Transport Layer Security

VIO Virtual Input/Output - Xilinx provide a debugging environment allowing Virtual Input/Outputs to be attached to hardware for debugging

XGMII 10 Gigabit Media Independent Interface

Chapter 1

Introduction

This thesis is the continuation of design work to create an FPGA reconfigurable co-processor card. It describes the hardware testing, hosting solution, firmware development and the higher software stack for enabling use of the ACE1 reconfigurable co-processor card. Much of the remaining introduction will be dedicated to familiarising the reader with previous work and establishing the starting point and scope for this work.

1.1 Description of previous work

In this section a short description of the ACE1 board will be provided as a summary and for the starting point of the thesis, hopefully without becoming overly verbose.

1.1.1 ACE Research Group

The ACE is a research group, based at the CHPC in Rosebank, Cape Town. Research is focused on a selection of interesting hardware platforms and their application to High Performance Computing (HPC). The microprocessor (CPU) industry has become increasingly parallel in the last three years and will continue to do so in the foreseeable future. The science community have, in recent times, become increasingly interested in using GPGPU processors to solve their computationally intense problems. Some novel CPU architectures like the Cell Broadband Engine and ClearSpeed processors have been used in commercial applications and could be used in HPC in a similar way to what we have seen with GPGPUs. The ACE lab research is directed towards matching hardware to software for achieving the best performance when solving computational problems.

1.1.2 ACE1

As per the title of this thesis, we have created the higher levels of the gateware and software stack, specifically for the ACE1 board. The aim of the ACE1 project is to create and enable a general purpose accelerator to allow academics to develop custom pipelines or processors for specific applications.

Gateware, from here on, will refer to any HDL written hardware that is loaded into one of the two FPGAs on ACE1, specifically for enabling one or more of the various hardware interfaces; this being distinct from the application solving hardware (synthesised from HDL code usually) and resources.

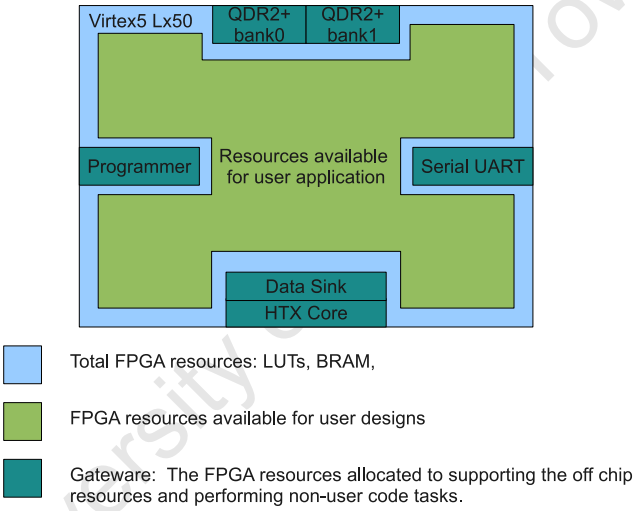


Figure 1.1: Differentiating user hardware from gateware. Note that, unless partial reconfiguration is used, both gateware and user code are combined into a single bitstream for programming the device.

The original hardware design and schematic capture for the board was done by Michael Aitken, with the actual board layout being outsourced. Manufacture of the PCB was then completed by StreamLineCircuits and the component placement done by Tellumat in Johannesburg. Finally, X-ray pictures were taken to ensure the placement of the BGA FPGAs. This work is an attempt to complete the ACE1 project by providing then necessary support level hardware (gateware) and low level software (drivers) for the host side of the system. Due to this work being a continuation of the design process it seems reasonable to provide substantial background on the ACE1 board and to clearly distinguish between the two separate efforts to create a functional and supported FPGA-based accelerator card. Section 3.2 deals with a summary of the board layout and the stage in testing that ACE1 was in when this work began.

1.2 Objectives

The overall objectives for the work to follow are outlined below:

- Test and assess the state of hardware interfaces
- Choose, procure and setup a hardware environment for ACE1
- Collect and design firmware, gateware and drivers for ACE1 interfaces
- Evaluate and try to optimize the gateware for lowest possible resource usage, such that as much of the device as possible is left over for user logic
- Provide an ecosystem to users allowing easy and reconfigurable access to all interfaces, including and most importantly, the FPGA programming interface
- A debugging environment was envisaged, much like the tools provided by the leading FPGA manufacturers, to allow test modules attached to synthesised hardware to return data over the primary board-to-host interface

As this is a follow-up project there is considerable detail in the introductory sections of [5] if readers wish to familiarize themselves with the overall objectives and motivation for ACE1 in general. The items above list only the initial requirements for this document.

1.2.1 Acknowledgement of Trade Marks

While working with an FPGA based co-processor board like ACE1 and the associated Intellectual Property cores that are associated with the various interfaces we encounter a significant number of companies, products and Trade Marks. We take this opportunity to acknowledge each of them, once and in alphabetical order in Table 1.1.

1.3 Chapter Summaries

Chapter 2 deals with the introduction to the co-processor model and background into the design decisions and metrics behind the creation of the FPGA-based reconfigurable co-processor board: ACE1. We outline the objectives, requirements and benchmarks required from ACE1 in order to become an effective tool in an HPC environment. We examine trends like Total Cost of Ownership (TCO) for hardware in data centers and how design decisions and metrics are changing with regards to power usage, management, support and assistance. We note that the cost of HPC hardware, when measured in FLoating point Operations Per Second

(FLOPS), is always decreasing, whereas the contribution to TCO from power, staff and cooling are ever increasing costs. The idea of heterogeneous computing is introduced and we examine how conventional processors have become faster due to more parallel cores, while not significantly faster in terms of serial performance over the last couple of generations. We discuss the successes of alternative architectures like GPGPUs coupled to CPU systems for improving performance and note how a versatile fabric like an FPGA can be used to implement custom cores for accelerating any particular application.

We examine typical FPGA usage and the fields where FPGAs currently see high usage outside of HPC. We compare these with some of the fixed architectures for suitability in an HPC environment. We note some very interesting examples in the industry at present of deals between the big players in CPUs and FPGAs. We also look into historical trends for co-processors in general and attempt to predict whether an FPGA co-processor is likely to follow a similar pattern to that of fixed architecture co-processors.

Lastly we discuss in some depth two other products that have come onto the market since the completion of design and manufacture for our reconfigurable co-processor board. We compare our co-processor system with what the market offers and note that we seem to have found an unfortunate middle ground position between two of the more effective HPC accelerators offered by industry.

Chapter 3 starts by outlining the board concept and layout including ACE1s various resources and interfaces and the reasons why they were chosen. We outline the programming interfaces and on-board memories which, in turn, determine how accessible the board is without any supporting gateware. The three devices on ACE1 that require configuration, namely: the two FPGAs and the platform flash, can be written to using JTAG and the platform flash allows the smaller FPGA to be configured at power-up in Master-Serial mode. In addition the smaller FPGA is able to program the larger - also in Master-Serial mode. We describe the technical background information needed to understand the gateware design and testing chapter as well as giving comparisons to some of the alternatives to each interface. We compare the performance of the various interfaces in terms of both bandwidth and latency and discuss the suitability of the on-board memory resources. The interfaces are separated into high-speed interfaces (listed below) and debugging and programming interfaces.

1. The board-to-host interface: HyperTransport via HyperTransport eXpansion,
2. The off-board interface 10 Gigabit Ethernet,
3. The on-board memory storage interface: Quad Data Rate 2 Random Access Memory.

We discuss the state of testing that was done on the board after manufacture, which effectively outlines the boundary between the end of the hardware design phase project and the start

of this project. Although industry tends to try and maintain a close integration between the hardware design and supporting gateware and low-level software, ACE1, as an academic project was not so closely coupled. Consequently there was a completion of the hardware design phase which involved some basic testing of the interfaces. We revisit the results of these tests.

A significant amount of time was spend arranging the hosting environment for ACE1 and the details of this are outlined in Chapter 3. Some of the issues that occurred with the hosting environment include: form-factor problems, power supply problems and boot time problems. The discussion covers going through multiple host machines due to both mainboards and ACE1 itself not conforming, or mistakes in implementation and how these problems were resolved. In the case of the problem of the edge connectors on ACE1 the solution required a printed circuit board to be laid out and manufactured in order to switch a set of incorrect connections. While it did correct the connection problems, it also interfered with carefully length-matched and impedance-matched tracks that the HyperTransport eXpansion standard requires. We review the implications of adding the converter card to the system.

The gateware designed for running ACE1 consists of cores supporting each of the interfaces outlined in Section 3.2. **Chapter 4** outlines the process of getting the ACE1 board interfaced with the host server, memory transfers, testing of cores, testing of the system as a whole and most importantly the boot procedure. We discuss the design of our management gateware which, when programmed into the smaller of the two FPGAs: v5lx50, would allow the host machine to write a bitstream to the board to be programmed into the larger v5lx110t FGPA. This requires the HTX interface between board and host to be up and running and to negotiate the link with the host machine at boot time. We discuss our efforts to support the interface using a core provided for supporting a similar HTX board made by the Center for HyperTransport Excellence based at the University of Heidelberg. This management core was seen as the crux of the problem to be solved; however it relied on the HTX interface to drive data from the host server.

The next sections in Chapter 4 discuss the details surrounding the core provided by the University of Heidelberg and our efforts in getting the ACE1 board and host server to communicate. We begin with the conversion of the core to support a Virtex-5 FPGA rather than the Virtex-4 based device that is used on the University of Heidelberg board, and discuss including the DMA application initially while testing. With the host machine still not recognising the board, we implemented a stripped-down version of the HTX core including only the initialization module, supporting buffers and clock management systems. In addition Xilinx Chipscope probes were added for tracking critical signals. With this simplified version we were able to debug the boot time issues but we will leave the findings and final summary of the problems with HTX for the sections that detail the major problems areas. They are found in Chapter

5.

With this in mind we show that a simplification of the management core does function, despite it no longer being relevant due to the problems with HTX. Due to the issues surrounding the boot-up and identification of ACE1 with the host system we switched our development efforts to supporting the networking interface attached to the larger v5lx110t FPGA as outlined in Section 4.4. 10GbE was only investigated towards the end of the project and, while it met with more success than the HTX portions, time constraints limited the development of the support gateware. We discuss the options that we provide for supporting the networking interfaces at the various layers of the OSI networking stack.

We provide suitable gateware for the ten Gigabit Ethernet interfaces on ACE1. The Physical layer is supported by the free Xilinx XAUI Coregen core and we provide some notes for users regarding changes to the schematic that have an effect on the gateware. We then provide two options for a Media Access Controller: one being a simplified version but at no cost and the other being the Xilinx 10GbE Media Access Controller along with an exerciser to test functionality.

In the OSI model the Media Access Controller in the Data Link Layer is usually the highest layer supported in hardware and higher layers in the stack tend to be run in software due to resource usage concerns. ACE1 does not have a hard microprocessor on it and we would have liked to have exploited the host CPU and operating system handling these layers. HTX would have the advantage of having low latency and the handling of the Internet Protocol and TCP headers is small and infrequent (relative to the data processing) but latency sensitive work. However, with HTX not available we provide an alternate option for using the 10GbE interfaces by implementing a soft-IP CPU (synthesised Intellectual Property based CPUs) and software stack for managing the MAC. Running a soft-IP CPU to support the higher layers of the OSI model is not ideal from a gateware perspective and there are options for cores that offer Address Resolution Protocol, Internet Protocol and User Datagram Protocol in hardware but these tend to be as bad or worse in terms of FPGA resource usage than a soft-IP CPU. We investigate an open hardware core that supports the upper layers of the OSI model but leave it to the user to decide how best to utilize the networking interface on ACE1. In a very simple case, frames could be generated and passed to ACE1 without any higher layer software/hardware. For a more complicated environment the options are to run a soft-IP CPU to control the Media Access Controller or the alternative being to use a completely hardware-based solution which would only implement a subset of the higher layer protocol features.

Chapter 5 highlights a common trend in hardware design projects. The first iteration a design often brings to light subtle (and sometimes not so subtle) issues and in many cases multiple iterations are needed in order to stabilise a system. Even then there are usually

improvements that can be made. We discuss the areas where ACE1 could be improved at a hardware level. Each of the issues can potentially be circumvented; however, this tends to create a number of further problems or complexities which should be avoided in subsequent revisions.

We investigate a problem that was noticed when ACE1 was powered from an external power source. Some of the power supplies are not properly isolated and the external power supply is driving some components on the mainboard. We trace the problem back to the power network and note that the 12V rails that form part of the HTX standard are properly isolated; however, the 3.3V rails that form the remainder of the power supply options in the HTX standard are not isolated.

The next and most substantial problem was the HTX non-detect problem which eventually halted development on the management gateware. We have discussed the work done to get HTX supported in previous chapters including creating a converter that sits between ACE1 and the host server and corrects a subset of the HTX signals.

There is a second issue that arises from the decision to have a dual FPGA daughtercard - how best to access the dual FPGA configuration of ACE1 from the host. HT can support both devices as separate nodes on the HT chain, however, we then need each FPGA to contain a HT gateware core and the core on the v5lx50 has to support tunnel functionality, which would have to be written and added to the University of Heidelberg HTX core (or purchased at significant cost). In addition, the second element in the chain would need to be hot-plugged, due to the bitstream for the larger FPGA having to come from the host via the smaller FGPA. This complication was not tackled due to us not being able to establish suitable communication with the smaller FPGA at boot time, however, we are confident that, with hot-plugging supported in HTX version 1.1, it would have been possible on ACE1. We argue that if we are implementing a gateware core to support HTX in the larger FPGA on ACE1 then it would make sense to have removed the smaller FPGA completely and relied on a partial reconfiguration solution to allow us the same reconfiguration benefits but with reduced complexity.

At the end of this section we outline the hosting costs of ACE1 as well as exploring some alternatives had ACE1 been reworked and become ACE2 during this project. We note the limited number of HTX enabled mainboards available, as well as a subtle configuration requirement on the Tyan based mainboards (and probably the other brands as well) where at least two of the CPU sockets need to be populated to gain access to HTX.

The work on ACE1 was concluded in **Chapter 6**, with work on the board-to-host interface abandoned and a late but still reasonably successful attempt at supporting the networking interface available on ACE1. The 10GbE provided hardware gives users a good place to start from to allow the board to communicate over the network. In this mode ACE1 would be

functioning as a stand-alone processing board. Time constraints prevented an investigation into shifting the soft-IP CPU (the recommended solution for handling the upper layers of the OSI networking stack) system from the v5lx110t to the v5lx50. Shifting the soft-IP CPU to the v5lx50 and creating a stand-alone processing board is the recommended direction to take for future work on ACE1.

We point out some corrections that should be made to the power network and the HTX interface (if the board-to-host inteface remains HTX) as well as the lack of flash for the v5lx110t. Finally we suggest re-working the PCB layout of ACE1 and create ACE2, as alternative to using ACE1 as a stand-alone board. We recommend, to future researchers wanting to build a second generation of our host-based co-processor board, attempting to use partial reconfiguration to remove the need for a second FPGA on the PCB and improving the programming interface.

Company or Group	product	description	symbol
Altera		Company name	®
Altera	NIOS	soft-CPU product	®
AMD		Company name	®
AMD	Fusion	CPU family	™
AMD	Opteron	CPU family	™
Apple-IBM-Motorola	PowerPC	CPU product	™
ARM		Company name	®
ARM	Cortex	CPU product	™
Celoxica	RCHTX-XV4	FPGA co-processor board	™
Dell		Company name	®
Dell	PowerEdge	Computer hardware family	™
HyperTransport	HT	A interconnection protocol for computer processors	™
HyperTransport Consortium	HTX	Socket based physical implementation for HT	™
Intel		Company name	®
Intel	Nehalem	CPU family	™
Intel	Atom	CPU product	™
Intel	Westmere	CPU family	™
Nallatech		Company name	®
Nallatech	BenONE-PCIe	FPGA co-processor board	™
Nallatech	PCIe-180	FPGA co-processor board	™
Nvidia		Company name	®
Nvidia	Tesla	GPGPU hardware family	™
PCI SIG	PCI	Serial point-to-point interface	®
Pico Computing	E-16 LX50	FPGA co-processor board	™
Tyan		Company name	®
Xilinx	ChipScope	Software product	™
Xilinx	Coregen	Software product	™
Xilinx	Microblaze	CPU product	™
Xilinx	Virtex	FPGA Family	™
Xilinx	MIG	Software product	™
Xilinx	Rocket-IO	Fixed hardware transceivers found in Virtex FPGAs	™
Xilinx/ARM	Cortex-A9	CPU product	™
Xtreme Data	XD2000F	FPGA co-processor board	™
Xtreme Data	XD2000i	FPGA co-processor board	™

Table 1.1: Acknowledgement of Trade Marks

Chapter 2

Literature, alternate products and industry state

2.1 Co-Processors and parallel architectures

The co-processor programming model is to off-load a computationally intense portion of an algorithm to a piece of hardware designed to handle that specific portion of the computation as quickly and efficiently as possible. This idea has been used and implemented in many systems and products over the years. One of the finest examples would be the Intel 8086 coupled with the 8088 mathematics co-processor. As the industry has improved its' ability to place increasing amounts of logic in silicon, as predicted by Moore's law parts of systems that were originally done as co-processors have moved onto the die area, vastly increasing our speed of communication with them [36, 49].

2.2 Parallel Speedup Laws and Limits

When solving computational problems using silicon technologies there are a number of resources available and a number of engineering trade-offs to consider. For each generation of silicon fabrication there is an ever increasing number of transistors on a silicon die that can be used to build processing elements and each can be clocked at a range of clock speeds. CPUs spent a number of generations using those additional transistors to improve pipelining and exploit instruction level parallelism. Improved performance was determined by: pipeline depth, how well the compiler and scheduler could keep a pipeline saturated and how fast we could clock that pipeline. Without worrying about the complexities of ever increasing pipeline depths; the cost of a higher clock rate is power and power is becoming increasingly expensive [2, 28]. The industry has run into some limiting factors as examined by University of Cali-

fornia, Berkeley in their 2006 paper The Landscape of Parallel Computing Research: A View from Berkeley [11]. In addition to the cost implications of high-clock-rate CPUs[1] in terms of power consumption, manufacturers reached limitations in our ability to dissipate heat from these devices. The industry has responded to these limits by expanding sideways and offering more CPU cores on the same die, at clock speeds in a more power conservative range. This change of direction for traditional CPUs introduced the need for writing parallel code in order to see improved performance on new devices. Parallel processing is not a new concept and the challenges regarding programming for highly parallel systems are well known. Historically the performance advantages of parallel implementations were negated by the long development times and the speed that serial processors improved. Due to the limitations outlined above we see current generation CPU products becoming more parallel and require parallel codes to fully utilize the device. In addition the problems that we are trying to solve computationally have both parallel and serial components so the advantages of having these additional CPU cores can be limited by Amdahl's law [10].

Amdahl's law states that the maximum speedup of a piece of code will always be limited by the portion of the code that is not parallelizable. His idea, originally specific to parallel processing elements, is often expanded into the concept that the slowest portion of the system will eventually cause a bottleneck as other areas of the system increase in speed. When we look at current computer systems we see:

1. Power is now a huge factor in the total cost of ownership for any computer system and therefore,

2. A single threaded CPU core that exhausts our ability to gain performance from additional instruction level parallelism will need to start communicating with other CPU cores if we wish to exploit thread level parallelism.

3. Memory interfaces need to be continuously updated to supply more data. If they are not intrinsically scalable then they will have to be re-designed every few generations.

There are some ideas that allow Amdahl's law to be bent:

1. Not all processing elements have to be the same

2. Not all processing elements have to be permanent

3. The problem size is not always fixed - Gustafson's criticism: if you run out of parallelism for a particular problem, make the problem bigger [21].

[1] Such as the Intel Pentium 4 CPUs

These are the advantages of heterogeneous computing and ACE1 aims to provide a platform for exploiting all of them in an effort to avoid some of the performance bottlenecks seen in the computing industry today.

2.3 Board Communication Costs

For any High Performance Computing (HPC) application benchmarking is a hugely important aspect of the project. When dealing with a traditional[2] compute cluster we are in an area of conflicting sciences. The compute cluster is a tool for solving problems (usually scientific in nature) but in addition it is also a research project in itself. Putting together the machines themselves is no mean feat and the design and implementation of these tools brings together a number of engineering fields. Apart from the design of the machine itself, additional issues can range from: power delivery and reliability, bandwidth delivery, cooling, infrastructure (buildings, security, support staff, etc.) and many others [24].

When analyzing the performance of these large compute clusters we see many different benchmarks associated with the machine. From machine hardware counts (number of CPUs/RAM/Network Bandwidth etc.) through to speed metrics for specific applications (usually measured in FLOPS achieved on benchmark suites such as LINPAC[3]) as well as metrics regarding power usage (performance per Watt is now a crucial data center metric). Whilst trying to avoid the traditional arguments about FLOPS, speedup and which benchmark is the fairest choice, we do need to focus on benchmarking for co-processors specifically. The ACE1 application accelerator board's key benchmarks are:

1. The communication cost of sending a task to the board.
2. The speed and ease with which data can be communicated between processing elements (eg. other ACE1 boards/data capture boards/other co-processing boards)

A fixed architecture co-processor is designed such that the processing element will solve the problem faster than doing it on a traditional CPU would. However with an FPGA co-processor the accelerator itself is programmable so we make the assumption that work done to implement a problem-solving system on the accelerator board would produce an unspecified speed-up for the task that is implemented[4]. However, we need to add the cost of getting the data on and

[2]We refer to traditional compute clusters as those made up of large numbers of x86 CPUs such as most of the machines found on the T500 list of supercomputers. There are a few exceptions, including a past number one on that list: RoadRunner, that contain processors like the Cell Broadband Engine that are early examples of heterogeneous computing. The majority of the list, as at January 2010, are full x86 based CPUs.

[3]LINPAC is the benchmark used to rank the top 500 supercomputers. It is mainly a test of floating point performance. `http://www.top500.org/faq/what_linpack_benchmark`

[4]It is up to the user of the co-processor to to ensure that the problem is suitable for implementation on an FPGA and that the code is optimised.

off the board into the equation before we can calculate a speed-up:

$$S = \frac{Tcl}{(T_l + T_{cc} + T_{ul})} \tag{2.1}$$

Where S is speed-up, Tcl is execution time of code on the local processor, Tcc the execution time for user code on the co-processor. The factors: T_l and T_{ul}, time to load data to the card and time to unload data from the card respectively, are key to the results expected from any co-processor system. It is therefore important that this cost, or penalty, for moving the computation to the co-processor is as low as possible. This particular benchmark influenced the design decision surrounding the two major ACE1 interfaces, as well as being a design requirement for the gateware:

1. The board-to-host interface - HTX. ACE1 had to have an extremely high-speed low-latency interface between the board and the host machine. Further discussion on the host-to-board interface for ACE1 is available in Sections 3.2.1 and 4.2

2. The board-to-board interface - 10GbE. An interface allowing ACE1 to be placed onto a network such that it could pull data in from outside sources as quickly as possible. As with any network, careful standardization is very important for allowing multiple devices to communicate efficiently. Sections 3.2.3 and 4.4 cover the background and implementation details for 10GbE on ACE1.

2.4 Reconfiguration Costs

A second feature that was selected for the ACE1 board was to have a dual configuration with the smaller of the two FPGAs (XC5VLX50-1ff676 - referred to from here as “the v5lx50”) having pins connected to the programming pins of the large FPGA (XC5VLX110t-1ff1136 - refered to from here as “the v5lx110t”) allowing the v5lx110t to be programmed by the v5lx50. This allows the ACE1 board to reconfigure the v5lx110t on-the-fly with the idea being to change FPGA hardware configuration during excecution of a substantial code (broken down into kernels). Working with reconfiguring FPGAs on-the-fly ties in with the techniques of partial reconfiguration for FPGAs. Partial reconfiguration is split into a separate section of the literature review, found in Section 2.11. HTX was chosen as the board-to-host interface due to the low latency and high speed that it provides, as shown in Table 3.1, such that we would have the best possible chance of allowing reconfiguration on-the-fly. Speedup must be computed as laid out in Equation 2.1 as a function of both the time it takes to move data around as well as the time taken to compute.

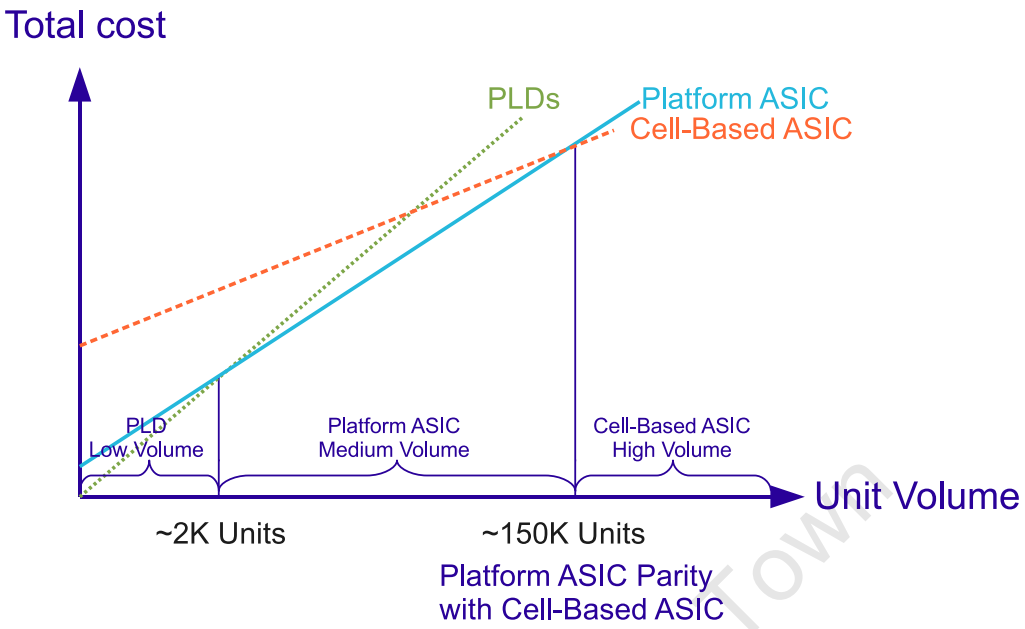


Figure 2.1: Graph of total costs against units for Programmable Logic Devices, Platform ASICs and Cell-based ASICs

2.5 FPGAs in HPC

Historically FPGAs have been used mostly in the embedded systems field. They are often seen as a prototyping environment for ASICs however, due to the high setup costs for producing ASICs, many systems are never upgraded past the FPGA. However the ability to apply hardware updates in a similar fashion to how we apply software updates can give FPGAs an appealing advantage over ASICs.

We see a significant loss in both power and performance when accomplishing tasks on FPGAs rather than using an ASIC for the job [31]. However, in a similar fashion to how the computer games industry drives the price of a GPU down; FPGAs can achieve better cost to performance ratios by exploiting the mass production of FPGAs for the embedded systems environment[5]. The unit price for both FPGA and ASIC based products is heavily dependant on volume, however with ASIC products we see much higher barriers to entry. A large unit number is needed when fabricating ASICs to push the Non-Recurring Engineering cost (NRE) component per unit down, however FPGAs are flexible enough that they are produced in high volume for a variety of customers and therefore a single custromer can still afford them even at low volumes. The FPGA manufacturers themselves offer tools to assist designers that are converting their designs from FPGA based to ASIC once the volumes and requirements suit the project.

The power consumption of these devices is a significant metric due to the traditionally very high power consumption of data centers. Project managers and planners involved with high

[5]although not to the same extent as we see in computer games industry

numbers of machines (be it for data centers or even IT shops managing large numbers of systems) are aware that the cost of running hardware for a little over a year will cost more than the purchase of that hardware [42]. While this is less true in South Africa where hardware costs tend to be higher and both staff and energy costs tend to be lower we note that Eskom, the South African parastatal electricity provider, had a tariff increase for the financial year 2008/2009 of 27,50%, and for the 2009/2010 financial year the tariff increase was 31,30% [1, 19]. The energy cost contribution to TCO, in South Africa, has begun to increase substantially.

We can save power and increase efficiency if we can exploit a co-processing or heterogeneous computing environment for solving computational problems. The ACE1 board aims to explore this space, specifically what an FPGA as a co-processor can contribute to HPC in terms of parallel processing, as well as lowering the power costs of computation.

2.6 Co-Processing

Co-processing has, in the past, followed the pattern:

1. A compute intense kernel is identified [6]
2. The ability to process this kernel is enabled or sped up significantly by adding hardware specific to the task
3. Over time, and as Moore's Law dictates, more transistors on the die enable us to bring this specific piece of hardware closer to the CPU by fitting it into the die area of the CPU. This improves the bandwidth available to the co-processing hardware significantly.

Some examples:

- The Intel 8088 Maths Co-processor
- Cache - used to exist strictly off-chip and has slowly migrated closer with huge portions of die area for popular CPUs now dedicated to cache. Multi-layered caches can still have some layers off-chip with devices like RAM Drives and more recently very high speed SSD drives however, as Moores law allows us more transistors, we see higher and higher percentages of them dedicated to on-die cache.
- TLS and SSL accelerators - Used in server systems that have to process high volumes of encrypted traffic like those that serve banking websites. A recent release in the media regarding the Intel Sandy Bridge architecture lists dedicated AES New Instructions (AESNI). AES is often selected as the encryption scheme for TLS and SSL.

[6] It has often been a real time processing requirement that drives us to step 2

- Graphics Processing Units - increasingly being used as General Purpose GPU (GPGPU) which we discuss in Section 2.7.

Within the industry we see a handful of other FPGA co-processing boards available and Section 2.10 contains a review of two of the products that are worth comparing to ACE1. In addition the HPC industry has begun utilizing some of the fixed architecture co-processors like the Cell Broadband engine and GPGPUs.

The ACE1 board was designed with the above trends in mind and we were keen to see it have as much communication bandwidth with the host machine as possible i.e. to place the co-processor as close (in speed terms) to the host CPU as possible. In addition, latency was a key issue with the intention being to allow us to hop data back and forth for as many small components of a compute kernel as the user desires i.e. fast reconfiguration on-the-fly. To expand on this idea; we wanted the use case shown in Figure 2.2 available to the end users.

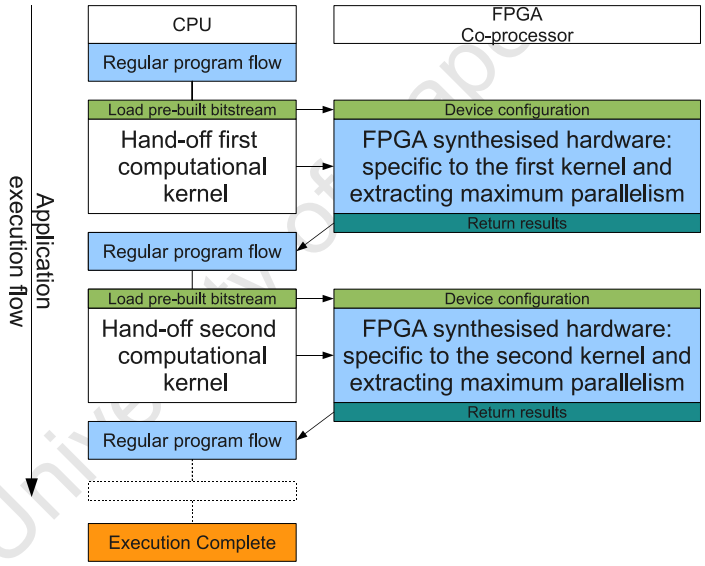


Figure 2.2: Code execution flow for a multiple bitstream application. Execution is shown in pale blue. Reconfiguration costs are shown in pale green and need to be minimised.

2.7 Discussion surrounding GPUs and more particularly the advantages of fixed architectures

There would be a serious deficit within this work if some time was not spent discussing fixed architectures and more specifically General Purpose Graphics Processing Units. When used for solving mathematical computations, other than shading pixels for games, we give them the "General Purpose" prefix. The GPU boards are fixed architecture co-processor boards dedicated to performing very high numbers of similar single precision mathematical

operations used for shading and texturing pixels. Both Nvidia and ATI now release GPU cards specifically for the HPC industry namely: the Tesla range from Nvidia and the Stream Technologies range from ATI. Both are making strides towards supporting double precision floating point numbers, mostly for the benefit of HPC. The Nvidia Tesla range is sold as a GPGPU: "designed from the ground up for high performance computing" - Nvidia [37]. The Tesla range is distinct from what most game players would purchase because they cost more than regular GPUs and contain additional components (like double precision floating point units) that are unnecessary for computer game graphics.

There are many papers available at present discussing remarkable speedups for scientific and commercial problems, using GPGPUs in everything from accelerating database queries to random number generation and molecular modeling [13, 40, 41]. When we look at the industry as a whole there is a situation in HPC where we find highly branched codes handled well by CPUs and certain mathematically intense kernels handled very well by GPGPUs. There are always a subset of codes that respond well when run on certain hardware. We have mentioned the University of California, Berkeley paper and subsequently an updated version was published in 2008, that has sparked significant interest in the HPC field in recent times [11, 12]. They take this idea one step further saying that by categorising all applications into Motifs (categories) based on their computational components we should gain insight into which applications would respond best to which processing elements. Many applications would require more than one Motif, for example; *the fast multipole method* for solving N-body problems would benefit greatly from co-processors that handle: particles, spectral methods and graphs-and-trees, and therefore possibly benefit from more than one co-processor [18, 3]. One significant advantage that an FPGA co-processor would have over a fixed architecture co-processor would be that reconfiguration could replace the need for multiple different types of fixed architecture co-processors being needed for a single problem or code.

Research into finding the best architectures for solving Motifs (originally referred to as Dwarf-hunting while the Dwarf term was still applicable) is an ongoing concern. Most of the Motifs are being matched to hardware that already exists, however, for the Motifs that don't find a good match with fixed architectures, FPGAs and ASICs are the alternative. The effort (and therefore cost) of creating a solution based on the above technologies goes up significantly as we move away from fixed architectures. CPUs tend to be comparatively easy to program, although it is becoming harder to improve performance when performance requires utilizing multiple cores. GPGPU programming tends to be more challenging than working with CPUs and once we move away from the fixed architectures there is substantially more difficulty in creating efficient programs and systems. Weighed against the difficulty in programming is the fact that we can gain improved performance (especially when measured per Watt) from implementing our systems on a co-processing element that best fits the problem. GPGPUs seem to be the sweet spot in the scientific community at the moment for ease of implementation

weighed against performance.

2.8 Following the co-processing trend

2.8.1 GPGPUs

With GPGPUs being the most exciting co-processing device in the HPC industry at present, we have been watching the progression towards closer integration with the CPU. As Section 2.6 suggests the likely trend is that the GPGPU will get progressively closer to the CPU. In 2006 AMD aquired ATI and the acquisition suggested that a product combining the two processing architectures was likely.

In August 2010 the first technology reports were released showing the details of the AMD Fusion [7] architectures: Bulldozer[8] and Bobcat. In addition, in September 2010 Intel announced their Nehalem/Westmere successor; the Sandy Bridge architecture, at the Intel Developer Forum 2010. Both of these new architectures show the first signs of a GPU-like processing element being included on the die alongside the CPU. "The processor family will include a new ring architecture that allows the built-in processor graphics engine to share resources such as cache, or a memory reservoir, with the processor's core to increase a device's computing and graphics performance while maintaining energy efficiency" - Intel Press release for Sandy Bridge architecture[9]. This seems to be very much in line with the trend that we have observed for past co-processors. The passage mentions energy efficiency as more relevant in the field of computing today than ever before.

2.8.2 FPGAs on the die (or FPGA embedded CPUs?)

FPGAs are seeing increasing use as co-processors which is what sparked the initial requirements and desire to construct and support the ACE1 board; we would like to create an open-source accelerator, at fabrication price, for the academic community following on from previous open-sourced accelerators [[5] Section 1.2]. In many cases the academic community are not made aware of the successful use-cases for these tools due to the sensitive nature of the work they perform in fields such as Radar (usually military Radar) and encryption/decryption. The FPGA itself is a very flexible tool and as each generation of hardware allows us more logic to work with we have seen the rise of popular soft-IP CPU implementations such as NIOS from Altera, Microblaze from Xilinx and also PowerPC, an Apple-IBM-Motorola product available for use as a soft-IP CPU on Xilinx FPGAs. These soft-IP CPUs have seen

[7] `http://sites.amd.com/us/fusion/APU/Pages/fusion.aspx`
[8] A review can be found at `http://www.pcper.com/article.php?aid=985&type=expert`
[9] `http://newsroom.intel.com/community/intel_newsroom/blog/tags/sandy_bridge`

widespread use in implementing the upper layers of the OSI networking stack. We discuss this further in our implementation of gateware for supporting 10GbE on ACE1 in Chapter 4.

A very recent development among the various FPGA players has been to associate themselves with one of the larger established CPU manufacturers and utilize already well established CPU architectures for building complete processing systems. Xilinx, in April 2010, announced an ARM Cortex-A9 processor-based platform and Altera have begun work on using the Intel Atom processor alongside their FPGAs [51, 7, 35]. The question of whether in future we will see parts of a regular CPU die contain reconfigurable sections remains to be seen, however with the flexibility of FPGAs we are already seeing tightly coupled CPU controllers in FPGA space.

We have discussed the sideways move that the CPU industry has taken towards having multiple similar cores adding significant parallelism to current processing systems. We are seeing moves to incorporate further parallelism in our processing systems by utilizing GPGPUs and other parallel fixed architectures. It would seem that the next logical step would be to include an even more parallel processing platform alongside our CPUs to suit the problems that can exploit those levels of parallelism. While this highly parallel platform would improve performance whilst keeping clock speeds down, programming such devices is notoriously difficult.

In the meantime FPGA based co-processors make excellent test beds for experimenting with handing off computationally intense portions of an algorithm to hardware designed and dedicated to solving that specific problem.

2.9 Power Wall

The Berkeley paper lists a number of issues surrounding HPC and refers to them as walls:

1. Power Wall - we used to design silicon devices without much regard for power, with our only concern relating to dynamic power. Now we are seeing more transistors on a die than we can afford to turn on and leakage power being as high as 30%.

2. Memory Wall - The memory available to CPUs has increased by a huge margin as our CPU speeds have increased. However with our ever larger storage capacities has come a penalty in terms of locating and fetching data. Load and store now takes in the region of 200 cycles[10].

3. Instruction level parallelism - CPUs are the best tool available to exploit instruction level parallelism (in combination with their very effective optimising compilers) and have been

[10]Without taking into account the complicated caching system that has developed to feed the modern CPU, the problem of memory wall exists despite our efforts to cache large portions of main memory.

refined over many years however, we are starting to see diminishing returns on searching for additional ILP.

The power consumption of a programmed FPGA can vary significantly depending on the application and its implementation, followed by the conversion to a bitstream by the manufacturer toolchain. For additional details of the boards total power consumption see Appendix A of [5].

2.10 Alternate products

In this section we review some of the similar and alternative products to the ACE1 board. When designing the ACE1 card these systems were reviewed ([5], Section 3):

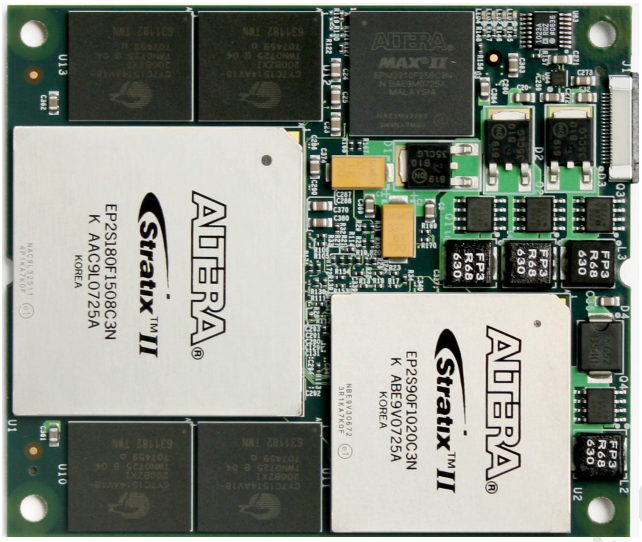
1. Celoxica's RCHTX-XV4
2. University of Mannheim HTX-Board
3. Nallatech BenONE-PCIe
4. Pico Computing E-16 LX50

Delving into the details of these co-processor boards a second time seems unnecessary, however in the three years since the design of ACE1 began, some other interesting and newer products have arrived in the FPGA co-processing marketplace. This section will review two of these boards and offer a few comparisons between each board and ACE1. Both are host-based FPGA co-processor boards and both support host-based re-programming over a fast interface; reconfiguration-on-the-fly.

2.10.1 XD2000[i or F]

The XD2000 series accelerator from Xtreme Data has two variants: the "i" and the "F", shown in Figures 2.3a and 2.3b. They are very similar co-processor boards both housing large Altera Stratix FPGAs. They lack the external interfaces like 10GbE that the ACE1 has, mainly due to the fact that they are buried within the host computer's chassis. When installed they reside in one of the CPU sockets (either AMD's socket F if its the XD2000F or an Intel Xeon socket system for the XD2000i) and by being in one of the CPU sockets they get the same bandwidth and latency that a CPU would have when accessing the systems resources. This avoids the need for the host machine to support HTX, as any dual socketed board (including the Intel boards for the "i" version) will work, and opens up the range of mainboards that can be used as a host machine massively, while maintaining the same performance as HTX would achieve.

The boards in the XD2000 series have similar on-board memory to ACE1: “36 MB of QDR memory”.



(a) XD2000f

(b) XD2000i

They are unable to provide 10GbE directly to the FPGAs due to limited PCB area as well as lack of access to the edges of the host machine chassis, however, these interfaces could be provided with a dedicated 10GbE NIC in the host system. Supporting 10GbE this way would add a significant burden to the communication channels on the mainboard (HT or Intel FSB depending on the platform) which would then impact on memory access via the mainboard.

This board is ACE1s closest competitor for latency and bandwidth between host and co-processor[11]. Judging by the market price (approximately $23000) for the XD2000 series boards there was clearly a lot of time needed for the research, development and support software associated with these boards. Working with and supporting these kinds of very high speed, often cache coherent, interfaces is not easy however they gain the advantage of FSB performance: “16 bits wide @ 800M Transfers/s” for the XD2000F and “FSB interface: 1066M transfers/s (8.5 GB/s data flow)” for the XD2000i.

[11]With the progression to HT3 and further versions of PCIe up to PCIe version 3 there are a number of faster interfaces now available. Future generation of these co-processor boards are very likely to exploit these resources as will any future versions of ACE1.

2.10.2 PCIe-180

The PCIe-180 is a product offered by Nallatech. It has a Virtex-5 LX155 user FPGA[12], PCIe 8 lane version 1.1, 512MB DDR2 SDRAM and a 10GbE port.

Figure 2.3: The Nallatech PCIe-180

We discuss the differences between HTX and PCIe across their respective versions and configurations in Table 3.1. The PCIe connection chosen for the Nallatech PCIe-180 board has a huge advantage over ACE1; it can be used with a far greater range of host machines. Just about every personal computer on the market would handle this card and the volumes and commodity nature of the host machine for this card would drive the price down to a fraction of the cost of our host machine for ACE1. This is a very significant factor when weighed against the original goals of ACE1, despite any loss of performance. Unfortunately Nallatech did not choose the option of a second 10GbE interface in their board design which we would consider a significant disadvantage when compared to the dual 10GbE interfaces on ACE1. The dual 10GbE ports would allow the board (and do in the case of ACE1) to sit as the middleman in a 10GbE network and handle high volume tasks on packets passing through it, such as encryption/decryption or deep packet inspection. The PCIe-180 boards cost approximately $6000 and a host machine for the PCIe-180 could be purchased at a very low additional cost. For a more in-depth look at the cost breakdowns of these systems see Section 5.6.

2.10.3 Comparison to ACE1

ACE1 sits exactly halfway between these two products in terms of design and interface decisions. It would appear that designers and customers who would like their products to cope with the stringent demands on the performance (latency more so than bandwidth) between

[12]Both major FPGA manufacturers offer sets of pin compatible FPGAs so often products have a range of FPGAs that they can use. The KAT project's ROACH board for example gives a choice of 2 pin compatible FPGAs: the Xilinx Virtex-5 XC5VLX110T-1FF1136 or the Virtex-5 XC5VSX95T-1FF1136. The PCIe-180 has an LX155 by default but other options are available with the same footprint as the LX155.

host and co-processor, have and will in future, opt for a co-processor board that fits a CPU socket. Those with slightly less demand will stick to the much easier and better supported interfaces like PCIe. PCIe, as can be seen from Table 3.1, still provides a huge amount of bandwidth. In addition newer FPGAs from both Altera and Xilinx come with semi-hard PCIe interface blocks within the FPGA to support this interface as quickly, and more importantly easily, as possible [8, 60]. They both provide example designs which enable designers to start from a known working point.

2.11 Partial Reconfiguration

We have discussed the ideas behind reprogramming the v5lx110t on ACE1 on-the-fly in Section 2.4 and we have noted for the products in Section 2.10 that this is similar to how some of the industry products function. They have a glue-logic device[13] that is inaccessible to the user and controls communication with the host and subsequently allows the user device to be programmed. In planning ACE1 the smaller v5lx50 FPGA was originally not intended to be an interface and control device exclusively. The argument against having any portion of the acceleration hardware located on the v5lx50 is that a significant development effort would have to be included in the user hardware to allow the two subsections of the design, split across the two FPGAs, to both contribute to the processing of the workload. While it does remain a possibility to have both FPGAs contribute to acceleration, the partition of the hardware would have to be decided by the user and we consider it unlikely that many users would accept the added complexity for the gain of the remaining unused logic in the v5lx50 after gateware is in place.

The v5lx50 resources are only likely to be used for glue-logic tasks and when considering the suitability of the FPGA for this task we note that a smaller, cheaper, faster to boot and preferably non-volatile device would be much more appropriate as a glue-logic device.

Partial reconfiguration has been a feature offered by FPGA manufacturers for quite some time however, due to the complexities inherent in the feature, as well as limited support in the toolchains, it has remained a fairly obscure feature. In recent times the major FPGA manufacturers have attempted to boost the use of partial reconfiguration by improving the toolchain support for the feature. The key element required for supporting partial reconfiguration is an easy to use and reliable communication boundary between fixed logic and dynamic partial re-configurable logic.

Partial reconfiguration can be an alternative to the approach of having a dedicated interface device and dedicated communication device. Part of the FPGA logic could be dedicated to

[13] Often this device is a CPLD rather than an FPGA due to its non-volatile nature and faster startup time. The trade-off is that CPLD devices have significantly less programmable logic available.

Resource	Virtex 4	Virtex 5	Virtex 6	Virtex 7
Logic Cells	41K - 152K	46K - 156K	128K - 476K	not yet released
Slices	18K - 68K	7K - 24K	20K - 74K	
CLBs	4608 - 16896	3600 - 12160	10000 - 37200	
BRAM (kb)	1728 - 6768	2160 - 8784	9504 - 38304	
DSP Slices	64 - 512	48 - 640	480 - 2016	
Serial Transceivers	0 - 20	12 - 16	20	
Select IO	448 - 768	480 -640	600	

Table 2.1: A comparison of the Virtex range of FPGAs. Each column is a summary of a family of devices showing the values of both the largest member and the smallest for each row(these are not necessarily the same device in each case. For example; the SX subset will have to highest DSP slices while having the lowest Logic Cells).

gateware and interface support and set to static (fixed) while the remainder of the device could be dynamically reconfigurable. This would eliminate the need for setting up a second FPGA on the PCB. While this is of limited interest to ACE1, due to the board having already been completed and manufactured, it is very relevant to work on ACE2. Partial reconfiguration could be used to move the complexity of reprogramming on-the-fly from the very permanent and complex structure of a dual FPGA PCB (or one FPGA and one CPLD on the PCB) and instead have the complexity located in a reconfigurable fabric, allowing problems to be ironed out in a hardware definition language.

2.11.1 Technology improvements in FPGAs

Having outlined a that partial reconfiguration solution could allow future versions of the ACE1 board to use only a single FPGA we provide some details of the improved resource counts in the newer Xilinx FPGA families. Both Xilinx and Altera have updated their FPGA range since work began on ACE1 and both are close to[14] releasing a second updated family (the Altera Stratix 5 and Xilinx Virtex 7). These new generation devices need to be considered for any future revisions of the ACE1 board.

It should be noted that Altera FPGAs can and should also be considered for future boards as well as other manufacturers who fabricate at the 40-nm or, when released, the 28-nm processes. We review the Xilinx range of families due to their direct comparison with ACE1.

As can be seen in Table 2.1; the Virtex 6 devices have improved by a substantial amount over their Virtex 5 equivalent. The smallest Virtex 6 devices are of a similar size to the largest of the Virtex 5 family. It is also clear that the increase was not as significant when moving from Virtex 4 to Virtex 5.

Section 2.10.3 mentions that some of the Virtex 5 and all except one (the xc6vlx760) of the

[14] As of First quarter 2011

Virtex 6 devices now contain hard-IP PCIe blocks. This dramatically improves the performance of the PCIe interface in a similar way to how DSP Slices improve complex mathematical operations. Also available is the option to configure the device over PCIe.

Chapter 3

ACE1 board and supporting infrastructure

After the very brief introduction to the ACE1 card from Section 1.1.2, this section of the literature will attempt to convey the state of the field of High Performance Computing (HPC) with specific respect to co-processors while ACE1 was being designed, laid out and manufactured. It includes a detailed summary of the ACE1 board itself in Section 3.2 as well as the hosting solution and supporting infrastructure. It concludes with the details of the design and manufacture of a converter card for correcting some of the problems discovered in the HTX interface, allowing ACE1 to be powered by the host server.

3.1 Concept and Ideas

As covered in Section 2.5 FPGAs offer the ability to have specific hardware configurations assigned to them. The idea to design an FPGA based co-processor board was not a new one and we have discussed some of the boards that contributed to the design of ACE1, as well as two of the co-processor boards that have come out more recently. We will expand on this idea in the following chapter as we look at ACE1 in more depth, but briefly; the ACE1 board should allow an optimal synthesized hardware module for accelerating a particular application kernel to be quickly programmed into the v5lx110t FPGA.

When using the ACE1 board to accelerate a complete code we expect to find multiple kernels (See Section 2.7) that each have a hardware implementation that optimizes performance[1] for the associated kernel. Kernels that suit implementation on FPGA fabric are those that are able to scale up to using the full logic resources on the device. In this case we may have

[1]The construction of these cores falls outside of the scope of this work. We are trying create and enable a general purpose accelerator to allow academics to develop custom pipelines or processors for specific applications where volumes are likely to be very low.

more than one kernel that could utilize the co-processor and they are likely to be dependent and unable to run in parallel. We would like to be able to switch out the kernel acceleration hardware on-the-fly (see Figure 2.2), this feature was designed into the ACE1 board. The time taken to update the FPGA system is critical if we intend to repeatedly swap out the hardware. A very slow re-program time would destroy any advantage that we may have gained by allowing multiple compute intense kernels to be consecutively written to the co-processor.

As we discussed in the benchmarking section, a specific property of benchmarking any co-processor system is to include the time it takes to offload data and instructions to the co-processor and we add an additional re-programming time to this benchmark. This original requirement contributed a significant portion of the overall ACE1 design including the choice of HTX as the board-to-host interface.

3.2 Board Layout

A top level overview diagram of the ACE1 board is shown below in Figure 3.1.

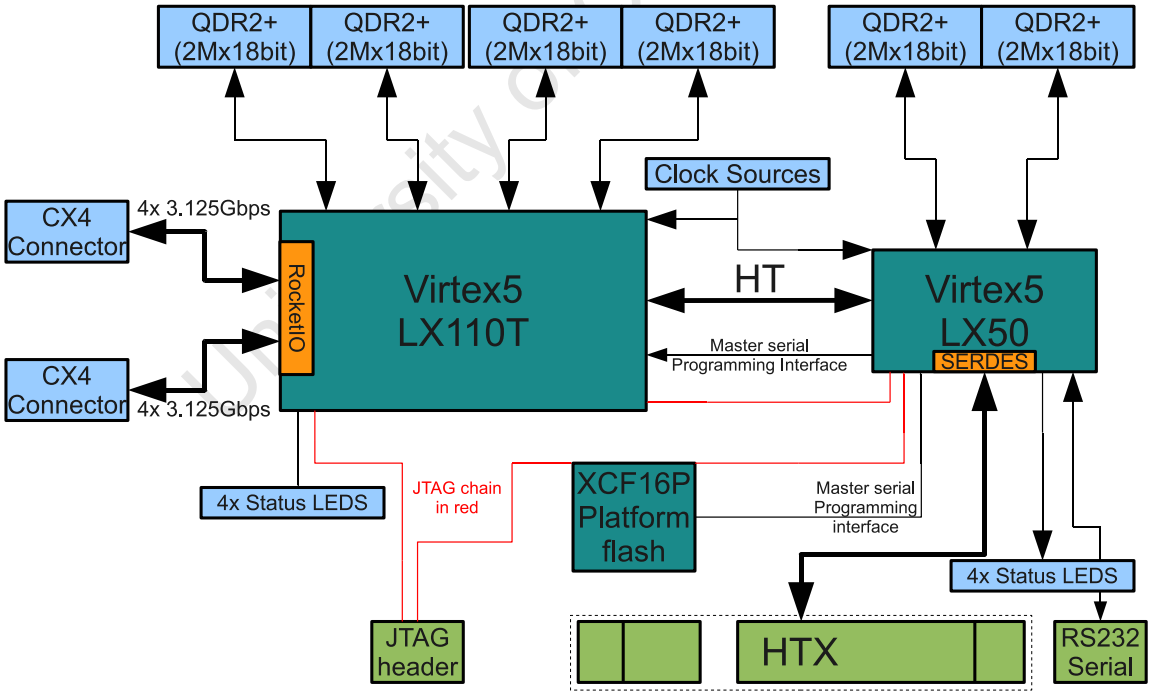


Figure 3.1: Top level overview of the ACE1 board showing its various resources and interfaces.

For the purposes of this work we will omit details regarding the power layout, the low level physical interfaces, the schematic capture and the manufacturing, except where they contribute to problems or advantages within this work. All of these items are covered extensively in [5]. Where we feel the reader would benefit from background knowledge is the choice of high speed and debugging interfaces, as well as their layout.

3.2.1 High speed interface: HTX

From the HyperTransport specification document we have the following introduction:

"HyperTransport technology is a high-speed, high-performance, point-to-point link for integrated circuits, and is designed to meet the bandwidth needs of tomorrow's computing and communications platforms.". As outlined in the benchmarking sections, the ACE1 board had requirements for a very high speed and low latency interface between host and co-processor board.

HT is currently in use as a replacement to the Front Side Bus (FSB) that has in the past linked the CPU to the North-Bridge chip and main memory. As with all buses we see that they don't scale well as the number of members on the bus increases [48]. In recent years we have seen the number of CPU cores on AMD and Intel chips increase substantially from one up to the latest chips that have twelve CPU cores[2]. AMD, in 2003, noticed that it was no longer the case that a FSB would connect the one or even two CPUs in a system to the system memory and peripherals. The FSB would soon need to connect many CPU cores to each other and to mainboard resources and we have already stated that a bus-based interface does not scale well. This particular problem has been solved before in computer networking; when multiple nodes all need equal opportunity to a communication medium there are very efficient protocols that allow us to share this resource. HT resembles a network protocol much more than a bus; it is packet based, allows for quality of service and prioritization, scales to high numbers of nodes and allows many CPU cores, often spread across multiple sockets, to share resources.

For clarity; the "X" portion in HT-"X" stands for eXpansion and refers to the slotted implementation of HT the protocol as shown in Figure 3.2.

[2]An example being the AMD Opteron 6172 Magny-Cours 2.1GHz 12 x 512KB L2 Cache 12MB L3 Cache Socket G34 115W 12-Core Server Processor

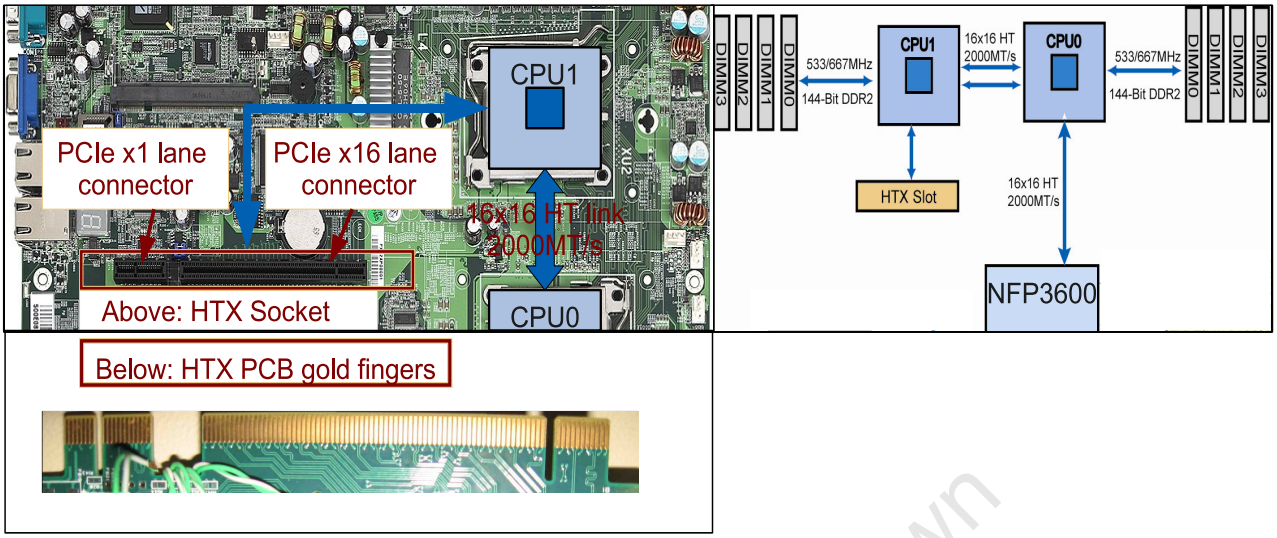


Figure 3.2: All HT links on the Tyan n3600R mainboard including the HTX connector needed for hosting ACE1.

At the time of ACE1's inception we had a situation of comparing PCIe v1 and HT v1.1 and in the time between our initial design and this publication we can see a few additional iterations of both these interfaces. A new entrant into the area of network-like FSB is Quick Path Interface used by Intel to support their CPUs. Table 3.1 provides a summary of the various interfaces, versions of each and number of lanes.

Version	Protocol	Lanes	Max. Clock Speed (GHz)	Theoretical max. unidirectional performance (GB/s)
1	HTX	16	0.8	12.2
1	PCIe	1	2.5	1
1	PCIe	16	2.5	8
1[1]	QPI	16	3.2	12.8
2	PCIe	8	5	8
2	PCIe	16	5	16
3[2]	PCIe	16	8	32
3[3]	HTX	16	3.2	48.8

Table 3.1: A comparison of the usual socket configurations found on mainboards and their associated speeds. Most of the underlying protocol (HT, PCIe) standards support up to 32 lanes and sometimes faster clock speeds when claiming maximum performance, however the mainboards don't provide implemented socket solutions for these cases.

Notes:

1. QPI is included due to it being a direct competitor to HT, however there is at present no slotted implementation of QPI to compete with HTX.

2. PCIe version 3 is currently (as at 3rd quarter 2010) still under development and is not available on any commercial boards. It aims to achieve double the performance of version 2, despite not doubling the clock rate, by changing the encoding scheme from 8b/10b to polynomial scrambling technique as well as 128b/130b encoding.

3. HTX version 3 is in a similar position to PCIe version 3. The specification, unlike PCIe version 3, is finalized and available however we are unaware of any products using HTX3 (as at 3rd quarter 2010) and more importantly any mainboards that support HTX version 3 daughtercards.

4. HT documentation from the HyperTransport Consortium appears to be the only set of literature with details for comparative latencies of the above interfaces [23]. Due to PCIe having independent lanes, and a single transaction having to complete over a single lane, we predict latency to be significantly higher than either HT or QPI, however documented results are scarce and somewhat one-sided.

Both QPI and HT allow for direct access to main memory and can have cache-coherent implementations. In a situation where we would combine many ACE1 boards into an HPC computing unit we would like to have moved to a cache-coherent HT core but finding a mainboard with a single compliant HTX socket proved challenging enough. As it stands testing on the ACE1 board was done using the HTX and a non-cache-coherent FPGA core. This is significant because of the way it is implemented, as a DMA based interface, which

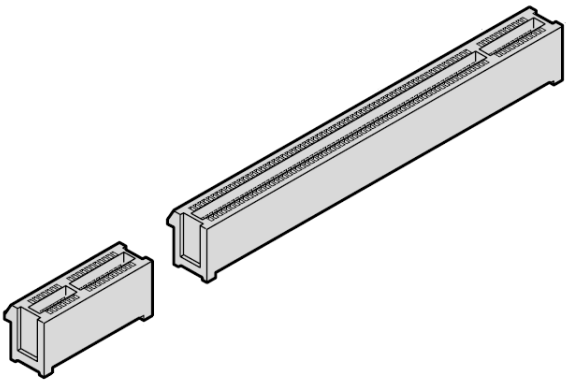


Figure 3.3: An HTX socket connector made up of two PCIe connectors, a 4 lane and a 16 lane.

makes it function in a similar fashion to how PCI and PCIe function.

Only due to its relevance further on in this document, the form factor for HTX is outlined here. The physical interface consists of an edge connector (gold-fingers) for the PCB side. The mainboard side, or socketed side, of HTX has a set of two connectors end-on-end that receive the PCB gold fingers. These connectors are the same as those used by PCIe; PCIe has a 4 lane standard supported by a 36 pin connector, as well as a 16 lane standard supported by a 164 pin connector. An HTX connector (see Figure 3.3) is made up using one of each of the 36 pin and 164 pin connectors. The physical interface consisting of socket and edge connector connects the host side signals, listed in Table 3.2, to the ACE1 side.

At the protocol layer the HT specification supports two types of packets: control and data; where control packets are either 4 or 8 bytes long and the specification ensures that control packets can interrupt long data packets to improve latency. Also to improve latency HT is designed as a parallel interface where a single packet is divided across the full width of the interface. Contrast this to PCIe where additional bandwidth is gained by banding multiple serial channels together to form a single high bandwidth interface, however a single transaction must still be completed over a single serial lane.

3.2.2 High speed Interface: QDR2+

The on-board memory to supplement that of the Block RAM (BRAM) and distributed RAM (made up of the reconfigurable lookup tables) internal to the FPGA was chosen to be QDR2+ RAM: 2M by 18bit wide RAM banks. The idea was that the card would have direct access to the host system's large Dynamic Random Access Memory (DRAM) based storage[3] via HTX and therefore a much faster and lower latency RAM (at the cost of size) would be selected for

[3]Most modern computers come equipped with at least 2GB of RAM, our host machine has 8GB

Signal name	Count	Direction	Level	Description	Required	
					Mainboard (MB)	Daughter Card (DC)
+12V	4	In	Power	12V power	Yes	Yes
+3.3V	3	In	Power	3.3V power	Yes	Yes
+3.3Vaux	1	In	Power	3.3V auxiliary power	Yes	Yes
VLDT	1	In	Power	1.2V HT power	Yes	Yes
GND	91	Common	Ground	Signal ground	Yes	Yes
CADIN[15:8][H:L]	8 pairs	DC→MB	LDT differential pair	HT data/command	No	No
CADIN[7:0][H:L]	8 pairs	DC→MB	LDT differential pair	HT data/command	Yes	Yes
CLKIN1[H:L]	1 pair	DC→MB	LDT differential pair	HT clock	Yes	No
CLKIN0[H:L]	1 pair	DC→MB	LDT differential pair	HT clock	Yes	Yes
CTLIN[H:L]	1 pair	DC→MB	LDT differential pair	HT control	Yes	Yes
CADOUT[15:8][H:L]	8 pairs	MB→DC	LDT differential pair	HT data/command	No	No
CADOUT[7:0][H:L]	8 pairs	MB→DC	LDT differential pair	HT data/command	Yes	Yes
CLKOUT1[H:L]	1 pair	MB→DC	LDT differential pair	HT clock	Yes	No
CLKOUT0[H:L]	1 pair	MB→DC	LDT differential pair	HT clock	Yes	Yes
CTLOUT[H:L]	1 pair	MB→DC	LDT differential pair	HT control	Yes	Yes
LDTSTOP#	1	MB→DC	2.5V CMOS	HT LDTSTOP	Yes	Yes
PWROK	1	Bidir[3]	2.5V CMOS	HT power OK	Yes	Yes
RESET#	1	Bidir[1]	2.5V CMOS	HT reset	Yes	Yes
REFCLK[H:L]	1 pair	MB→DC	3.3V diff LVPECL[2]	200MHz HT reference clock	Yes	Yes
SMCLK	1	MB→DC	3.3V CMOS	SMBus clock	Yes	No
SMDAT	1	Bidir	3.3V CMOS	SMBus data	Yes	No
TCK	1	MB→DC	3.3V CMOS	JTAG clock	No	No
TMS	1	MB→DC	3.3V CMOS	JTAG mode select	No	No
TRST#	1	MB→DC	3.3V CMOS	JTAG reset	No	No
TDI	1	MB→DC	3.3V CMOS	JTAG data in	No	No
TDO	1	DC→MB	3.3V CMOS	JTAG data out	No	No
USER	2	User	User	User-defined	0	0
RSVD	4	0	0	Reserved, do not connect	0	0
RSVDFS	6	0	0	Reserved, future standardization	0	0

Table 3.2: Table of signals for HTX link

the on-board storage. We would like the co-processor board to have options available similar to the caching structure of a CPU. The FPGAs themselves include the fastest but smallest storage regions (registers and BRAM), followed by the QDR2+ and then the main system memory over HTX. The interface to QDR2+ is well supported by the Xilinx generated QDR2 core. Further work on this gateware would be to allow the QDR2 to act as a cache for the host server's system memory available over HTX.

3.2.3 High speed Interface: 10GbE

The IEEE 10Gigabit Ethernet (10GbE) standard is an open standard which supports both copper and fiber links. Like its predecessors in the Ethernet family it is well supported by industry with 10GbE switches readily available, albeit at substantial cost, from the key companies in the networking field. The choice to use this interface was reasonably straight forward. The decision to go with copper rather than fiber came down to costs, with copper being substantially cheaper, and it was a 10GBASE-CX4 standard that was selected for the physical layer on ACE1. 10GbE is already well utilized in the HPC industry as well as on a number of other co-processor boards. As can be seen in Figure 3.1 ACE1 has two 10GbE connectors.

3.2.4 Debugging and supporting interfaces

The debugging interfaces on ACE1 are fairly limited. The v5lx50 has an RS232 interface connected to pin headers on the board via a MAX3388E PHY device. RS232 is a common method for communicating with a soft-IP CPU system or gaining shell access to an embedded operating system. The MAX3388E device allows for dual send and receive ports, however, both sets are connected to the v5lx50. While there is the option for allowing the signals to be passed through the v5lx50 and on to the v5lx110t doing so utilizes at least two (four for both channels) of the HT lanes making them unavailable for HT itself.

The other option for debugging is Chipscope[4] which runs over JTAG and Xilinx provide hardware modules that act as Integrated Logic Analyzers (ILA) and Virtual Input/Outputs (VIO) and can be attached to HDL hardware modules. We provide a reference example to familiarize users with the Xilinx debugging tools. The example project can be found in Appendix A.

[4]Xilinx require a valid ISE licence for using Chipscope, the use of Chipscope is not covered by the webpack (free) licence.

3.3 Programming ACE1

Xilinx provide a number of options for programming their Virtex series of FPGAs. The various options can be found in Chapter 2 of the Virtex-5 FPGA Configuration User Guide and the options available for programming the two FPGAs on ACE1 are highlighted in the following subsections [53].

3.3.1 Xilinx platform flash - XCF16P

The Xilinx platform flash is a non-volatile storage device that is programmed via JTAG to persistently store an FPGA bitstream. When the board is powered up the image in flash can be loaded to the v5lx50, depending on the settings of the programming interface jumpers (See the ACE1 schematic[4], v5lx50 on page 6 and v5lx110t on page 20, select settings according to Virtex-5 FPGA Configuration User Guide, Table 2-1 [53]). The XCF16P and v5lx50 devices can support parallel configuration mode (at the cost of additional pins) however this feature was not selected during ACE1 layout so the options are for Master-Serial or Slave-Serial modes only [54].

3.3.2 Virtex-5 Lx50

The smaller of the two FPGAs on ACE1 can be programmed from the Xilinx platform flash at boot time in Master-Serial mode or via the JTAG chain. JTAG does not give us the option of loading a bitstream at power up; it is used for live programming and for debugging. It requires a manual boundary scan from the host machine followed by either manual selection of a bitstream or a script to be run by the Xilinx Impact tool. It takes a similar amount of time to program the board over JTAG as it does in Master-Serial mode.

Master-Serial mode implies that the internal FPGA configuration logic is driven by a clock sourced from the CCLK pin which is generated by the Xilinx platform flash. Using this clock the data is read from the platform flash. In this mode the time to program the v5lx50 from the platform flash is approximately 6.5 seconds. Using Master-Serial configuration modes were simple and saved on IO pins but they are also a very slow method of programming the devices. The slow configuration time on the v5lx50 didn't influence our plans for reconfiguration on-the-fly, however it did cause problems when trying to have ACE1 identified by the host server at boot time.

Due to the fact that these interfaces don't share any pins they can be used in tandem without interfering with each other. The jumper configuration required to set the board to load an image from flash at boot time can be found on page 6 of the ACE1 schematic [4]. This image

can be overwritten via JTAG later on if desired without needing to adjust any of the jumper settings.

3.3.3 Virtex-5 Lx110t

The larger of the two FPGAs on ACE1 does not have its own platform flash and there is not enough space on the XCF16P flash to accommodate both images, therefore the v5lx110t cannot take on an image at power-up time. The v5lx110t does however have programming pins connected directly to the v5lx50, also in Master-Serial mode, and therefore we can load an image into the v5lx110t from the v5lx50. This was done in an effort to enable programming-on-the-fly as discussed in Sections 2.4 and 2.11. We used one of the SERDES hardware blocks that are found at the edge of the Virtex 5 FGPAs to drive the Master-Serial clock at the highest possible rate. The Virtex 5 SERDES support up to 600MHz DDR rates however we would be unable to use DDR while programming the v5lx110t from the v5lx50 using Master-Serial. Therefore 600MHz would be the fastest that we could program the v5lx110t and gives programming times of around 0.0064 seconds[5] or 6.4 milliseconds. While millisecond programming times are reasonable for FPGAs (where programming time is not usually a major factor in most designs) they still create a major bottleneck when using the reconfiguration on-the-fly techniques that the board enables.

The v5lx110t also forms part of the JTAG chain and therefore can be programmed via JTAG.

3.4 Testing of Physical Interfaces

When work on this project began the ACE1 board had not yet been manufactured. Snippets of code were being developed and collected while the board layout and manufacturing process were being completed. The state that the board was in when it was received is presented in this section.

3.4.1 HTX

HTX is separated into its two distinct interfaces:

1. HTX between host and v5lx50 and
2. HT between v5lx50 and v5lx110t

From [5], specifically: 7.2.3 we have: “To run the link at a faster clock rate [faster then the 100MHz that the HT links were tested at], delay matching using delay matching primitives

[5] A total of 3889966 bits need to be transmitted to configure a v5lx110t. Programmer overhead is negligible.

within the Virtex-5s would be required. A HT Controller core would be expected to implement this delay matching and provide a calibration mechanism.". In absence of this controller core (gateware) the HT interface between the v5lx50 and v5lx110t was tested with a simple loopback mechanism and driven at 100MHz. The HTX between host and v5lx50 was not tested prior to commencement of this work.

The gateware associated with enabling HTX between the host and the board constitutes a significant portion of this work and is discussed in Section 4.2.

3.4.2 QDR2+

The QDR2+ interface was generated using the Xilinx Coregen software and tested to the point where the core reported calibration for each of the QDR2+ modules at 150MHz, with the exception of bank three which only managed 100MHz. This detail is laid out in Section 7.2.4 of [5]. Additional work in integrating and allowing access to the QDR2+ banks can be found in Section 4.3.

3.4.3 10GbE

Details on the testing done on the 10GbE interface can be found in Section 7.2.5 of [5]. The following quote sums up the testing for this interface: "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.". Essentially the ACE1 was received, at the start of this work, with the 10GbE interface tested only at the PHY layer[6]. Additional work on accessibility and usability of 10GbE is detailed in Figure 4.6.

3.5 Host Machines

As a first step to developing gateware and low level support software for the ACE1, a host machine had to be procured. The choice of HTX as a board-to-host interface now limited the selection of this machine to just a single choice. The only option available in South Africa at the time was the Tyan n3600QX. This machine was placed on order from RedLynx SA while the first two ACE1 boards were being manufactured by SteamLineCircuits and the component placement subsequently being done by Tellumat.

[6] An understanding of the basic OSI stack for networking is important for working with any of the Ethernet standards including 10GbE. We outline the stack in Figure 4.6 but more detail can be found at `http://en.wikipedia.org/wiki/OSI_model`

3.5.1 Tyan's n3600QX mainboard

The n3600QX is sold as "4-socket server solution" by Tyan and Figures 3.4 and 3.5 show the board as well as a block diagram overview. The n3600QX is designed to be hosted in a flat 1U or 2U rack-server chassis. Due to ACE1 not being a low-profile daughtercard it was necessary to find an upright chassis for the mainboard. The delivery of the mainboard was stalled while wasting time looking for an upright (tower) chassis for the rack-server mainboard; going as far as trying to bring one over from Taiwan. Finally it was decided to run the board without a chassis and delivery of the mainboard and components was completed within a few days. The major components of the host server are outlined below:

1. The n3600QX mainboard with HTX connector
2. An AMD Opteron 8350, with heat sink and fan
3. 16GB of RAM for main memory (8x 2GB configuration)
4. Power Supply Unit: 850W
5. Sundry components: HDD, CDROM, etc.

Upon the arrival of the n3600QX it was immediately obvious that there was a hardware issue that would further delay the testing of ACE1 (as it turned out a substantial delay) in a host-based environment. The series of Figures 3.6, 3.7 and 3.8 show any hardware designers worst case scenario of a form factor error between the edge connector and the socket. The device with the error had the smaller of the two connectors (the 4 lane PCIe connector) that make up a full HTX connector inverted with respect to a vertical axis.

With this problem immediately obvious, work began combing the datasheets and specifications documents to determine which side of the interface was the problem. When it was established that ACE1 seemed to comply fully with the HTX specifications (in form factor at least) it was clear that the n3600QX was at fault and a ticket[7] was opened with Tyan itself. Going directly to Tyan turned out to be an error as they were not particularly helpful, responding only with a document titled: "TYAN Compatibility Test Plan/Report" [43][8] with subsection: B13. HTX Devices Compatibility Test and only after a substantial delay. This lists only that one of the cards made by Pathscale supposedly with a model name/number: HTX/HT Link1 has a rating of: "PASS". We were aware of Pathscale making an HTX-based card, shown in Figure 3.9, however it would seem from this figure that the Pathscale card would also not fit this board.

[7]The term that Tyan, and various other technology companies like Xilinx, use for a support query submitted via their web based technical support system.

[8]The document is included in Appendix A

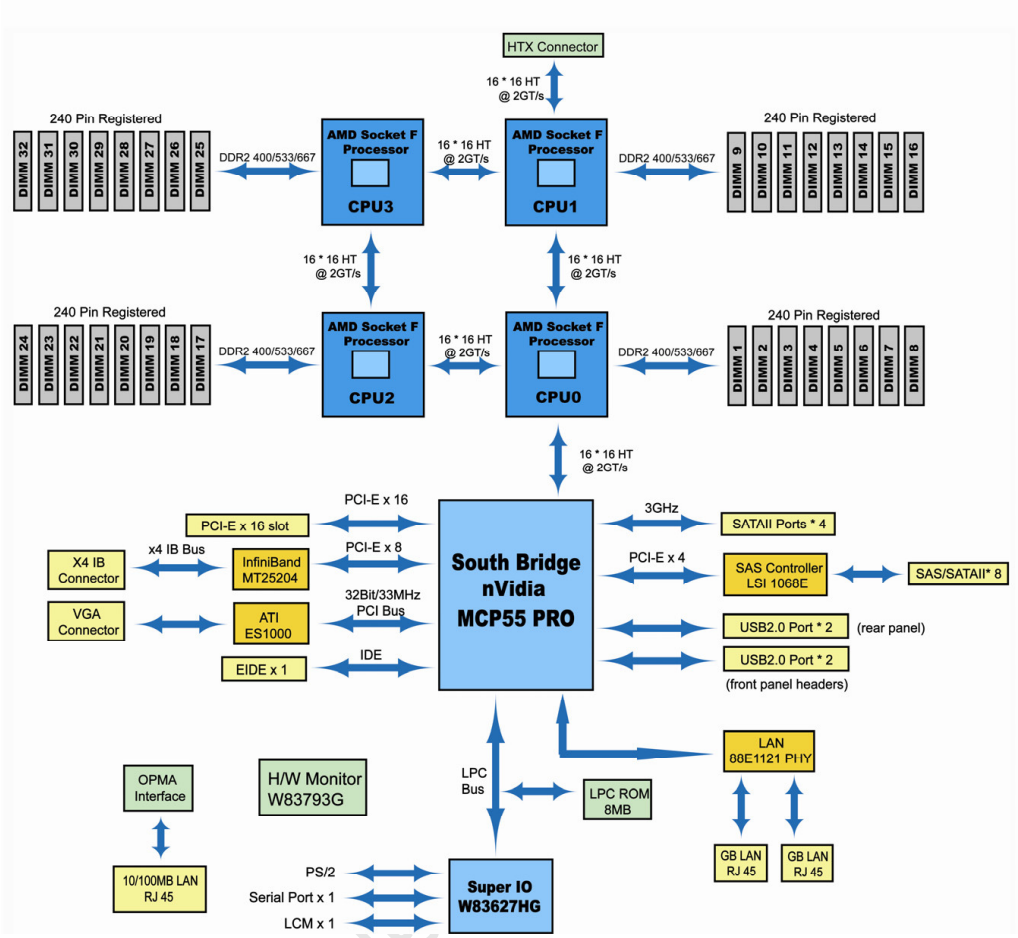


Figure 3.4: Tyan n3600QX mainboard block diagram

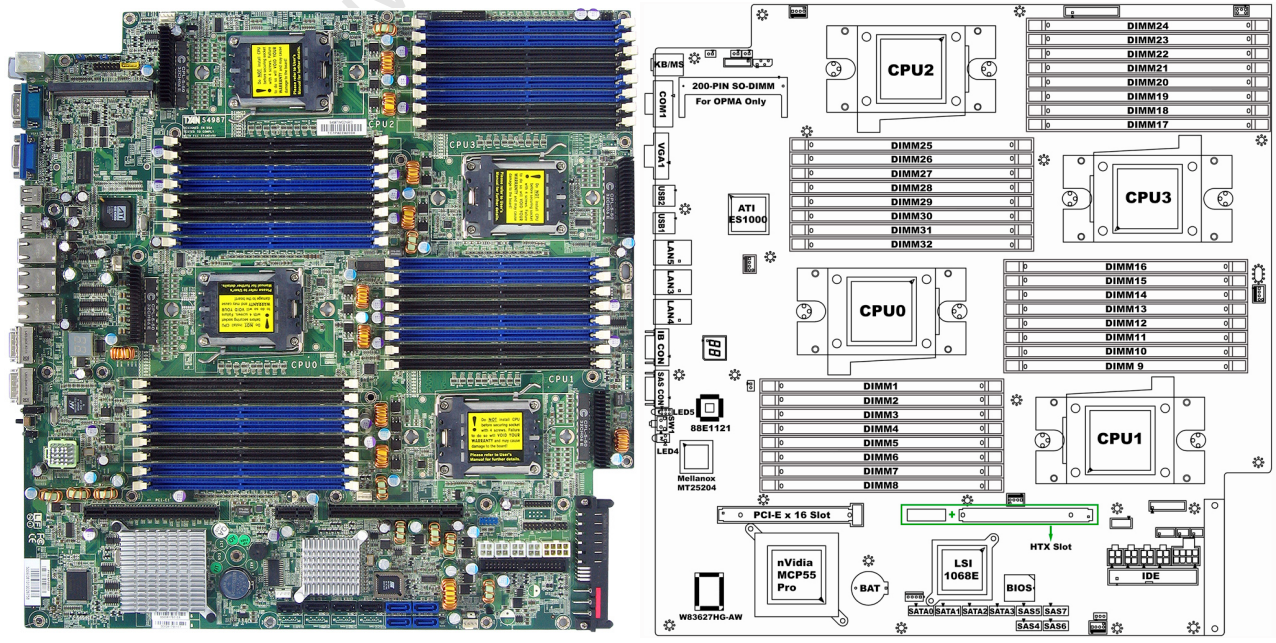


Figure 3.5: Tyan n3600QX mainboard showing interfaces and devices diagram showing HTX

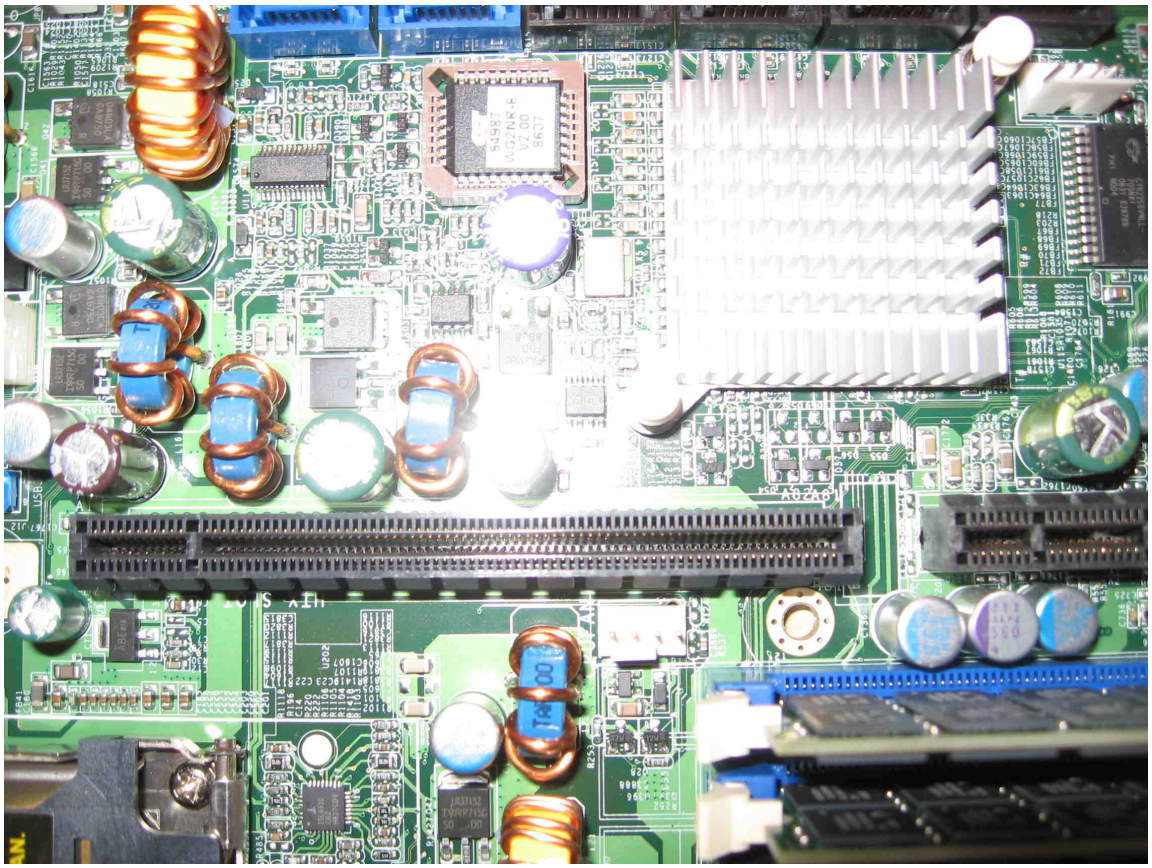


Figure 3.6: Close-up of the HTX socket on the Tyan n3600QX

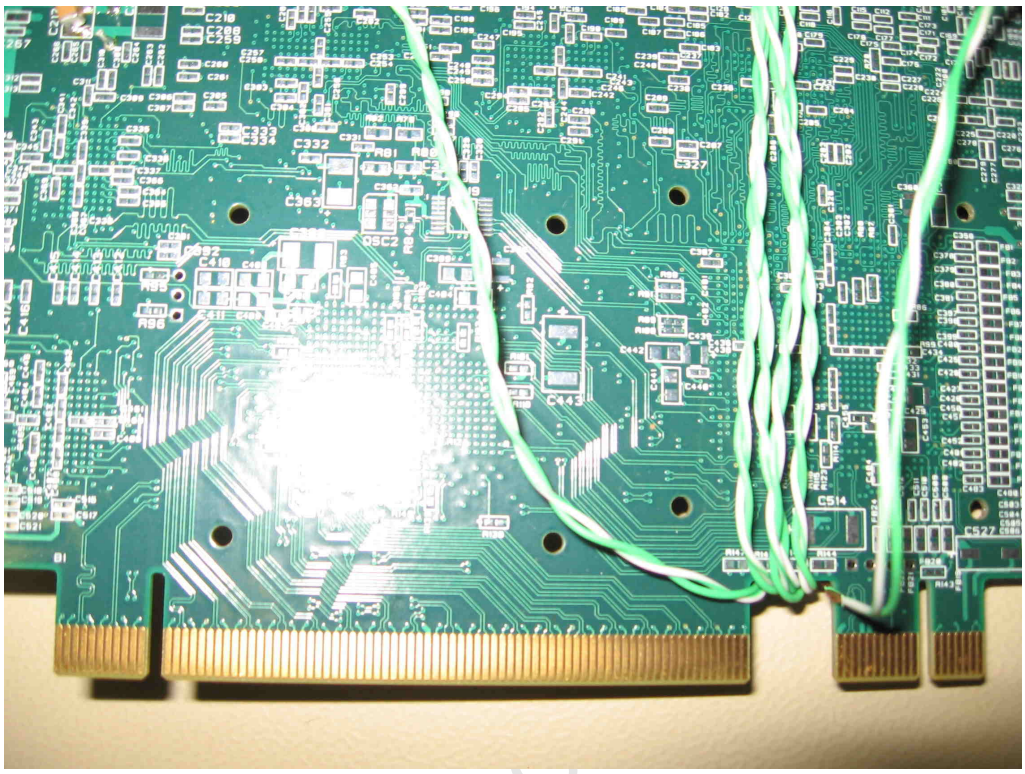


Figure 3.7: An image of the ACE1 board power prototype (unpopulated) showing the edge connector (gold fingers)

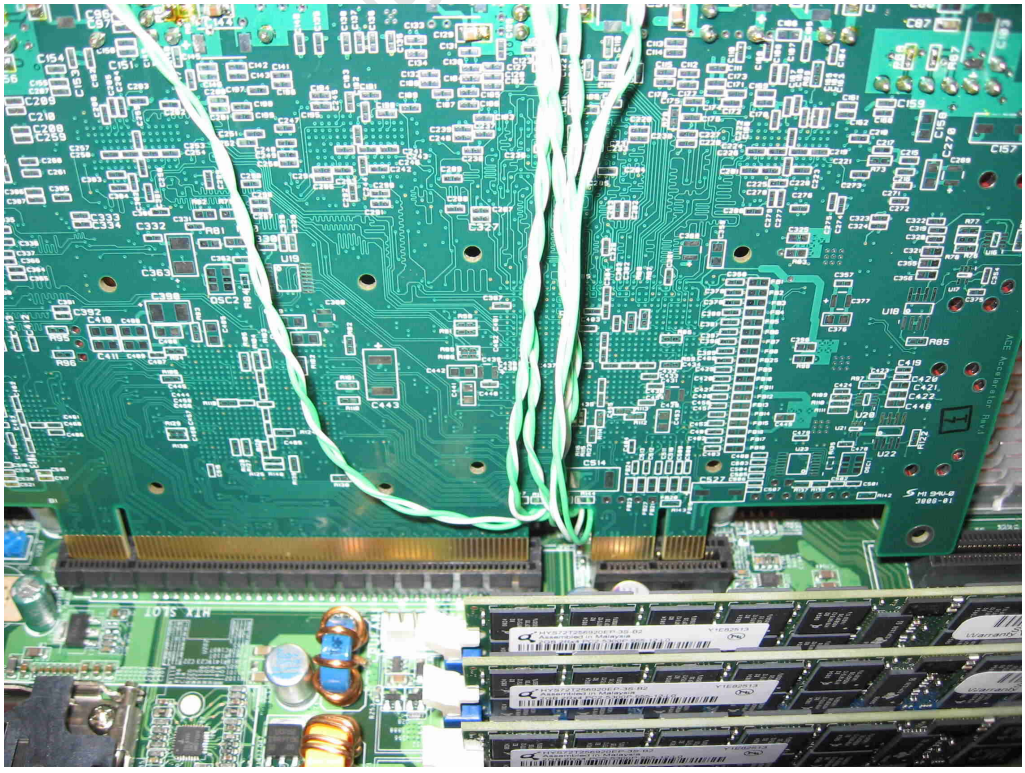


Figure 3.8: The form factor mismatch between ACE1 and the n3600QX's HTX slot

Figure 3.9: Pathscale HTX daughtercard showing HTX edge connector for comparison with the n3600QX socket shown in Figure 3.6

Figure 3.10: The front and back view of the riser card for the n3600QX that inverts part of the HTX connector.

When contacting the local supplier, RedLynx, a lot of effort was put into working with us towards a solution. Firstly a riser-card was found by RedLynx and brought to us for an initial test. This riser-card seemed to correct the issue by flipping the smaller of the two connectors, that make up an HTX socket interface, to the correct orientation. Unfortunately the ACE1 board was never powered up through this riser card as it did not belong to us and the riser was not available for purchase (no reason given by the supplier). Instead Redlynx proposed to exchanged the n3600QX for the n3600R, a newer mainboard that was considerably more suitable to this project and not available at the time that the n3600QX was ordered.

Due to the fact that the n3600QX is a rack-server motherboard it is understandable that a riser-card would be used when connecting daughtercards (over any interface like PCI,PCIe,HTX) due to the vertical space in a 1U or 2U rack-server chassis. When a riser-card is used for daughtercards the card would then lie parallel to the mainboard and would fit into the server chassis. What is unclear is why Tyan would use this riser-card to flip the small four lane PCIe connector that makes up half of the HTX connector, rendering the HTX socket on the mainboard unusable without the riser-card. It adds complexity to what could be a much simpler riser-card and creates additional work length-matching the tracks on the board.

3.5.2 Tyan's n3600R mainboard

The n3600R is a much smaller board than the n3600QX; designed to house two AMD Opteron CPUs connected via HT and with the availability of an HTX connector. This board, perhaps due to the fact that it is not designed for a rack-server, used the correct form factor for the HTX connector with no need for a riser-card or converter. We were able to house the mainboard in a regular chassis, install the ACE1 board and begin powering up our co-processor.

Our host machine used all of the original parts from the n3600QX now neatly packaged in a regular desktop chassis:

- The new n3600R board with HTX socket
- The dual AMD Opteron 8350 with heat sinks and fans
- The same 16 GB RAM
- The same Power Supply Unit: 850W
- The same sundry components (HDD, CDROM)

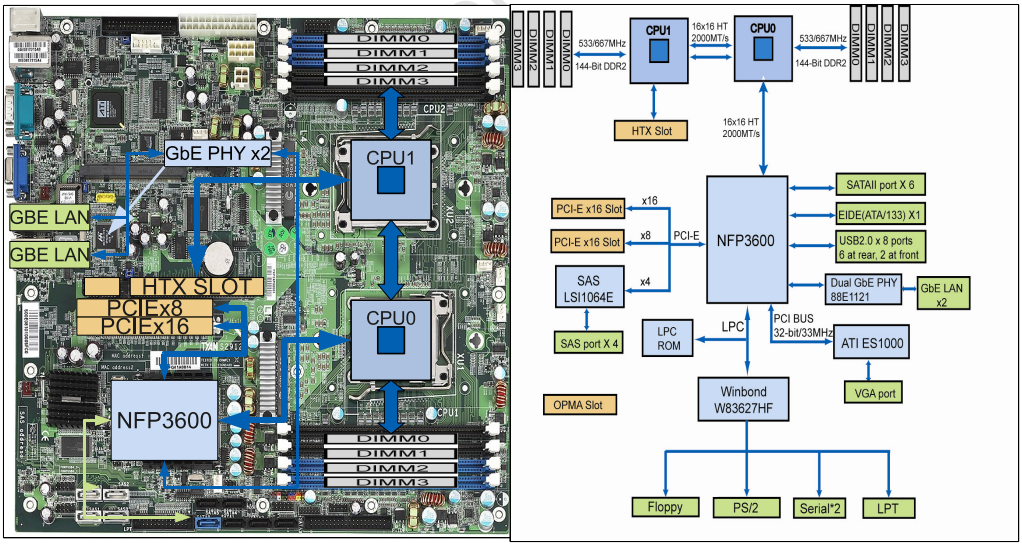


Figure 3.11: Tyan n3600R mainboard and block diagram showing all interfaces

One additional issue that cropped up with the procurement of our host system was the CPU cooling system. As per the list above we had a heat sink and fan for our Opteron CPU however, the new mainboard had an unusual mounting system. AMD have "socket F" specification for housing their 1207 pin CPUs which is well known and well standardized. In addition the standard includes two distinct mounting systems for CPU coolers; the common 3.1 inch mounting pitch as well as a less common 4.3 inch mounting pitch (used only by Tyan as far as we are aware). The initial cooling system that was provided with the Opteron CPU was

incompatible with the mounting points on the mainboard[9]. After much correspondence with RedLynx and a member of their technical staff coming out to check that we were not making a mistake, the two systems were declared incompatible. The cooling system was returned and a pair of coolers suitable for 4.3 inch mounting pitch were ordered (one spare in case of failures). Unfortunately the only supplier was based in the USA and once again we experienced delays while waiting for parts from overseas.

Upon arrival the cooler was installed and the host machine booted up, an OS was installed (Ubuntu 8.04, the most up to date Long Term support package at the time). The choice of OS will be discussed further in Section 4.2.5.

3.6 ACE1 socketed

With the host machine up and running, work commenced on powering up the board from the host system. Testing began with a few checks such as that the voltage rails supplied by HTX for the 3.3V and 12V rails were correct. It was initially decided to run the board using the additional 12V supply from the host machine PSU (referred to from here as: PSU12V) rather then drawing power from the HTX interface. These power supplies are often used when daughtercards require additional external power (usually GPUs). Our 850W PSU, as mentioned in Section 3.5.2, is more than capable of supplying power to both the host machine and ACE1.

When ACE1 is powered off the PSU12V but not connected to the host, both the mainboard and ACE1 power up as expected. The ACE1 loads an image from platform flash and initial tests aimed to determine whether ACE1 can be powered from the PSU12V connector.

When connecting ACE1 to the host server via HTX and attempting to power up the two systems concurrently, there were problems with both the external PSU12V power supply and the HTX socket power supply. The host PSU begins powering up and we see the signal LEDs on the mainboard light up and system fans are powered and start spinning. However during the power up process ACE1 causes a reset to the host power infrastructure which then attempts to restart the power up sequence thus creating a cycle. The various power-up behaviour problems were documented with videos that can be found in Appendix A. The host system PSU was unable to maintain the required voltages with ACE1 attached. This was initially attributed to the capacitance that ACE1 adds to the system but before attempting to remove the large 1000µF capacitor from ACE1 and replace it with a smaller version, the edge connectors were re-checked.

In addition to the reset cycling problem there was a second more subtle problem that was

[9]The Zalman 9700 series cooler is designed for the 3.1 inch mounting pitch. `http://www.zalman.co.kr/Eng/product/Product_Read.asp?idx=164`

noted. In the case where ACE1 was powered externally, and was in a position where it was powered while the mainboard was off, ACE1 was leaking power onto the mainboard. The signal LED for HTX on the mainboard could be powered through the ACE1 board. This problem is discussed further in Section 5.1 and we return to debugging the edge connectors.

A careful check of the edge connectors revealed a problem between the HTX socket on the mainboard and the ACE1 edge connector signals. The Figures: 3.12a and 3.12b, outline the signals that run through the smaller of the two PCIe connectors that make up the HTX connector and we note from this comparison that the edge connector was inverted by either Tyan or ACE1. When consulting further documentation we found that the fault was on the ACE1 side of the interface [16].

Pin	A	B	Pin
1	USER2	GND	1
2	GND	REFCLKL	2
3	GND	REFCLKH	3
4	RSVDFS	GND	4
5	RSVDFS	GND	5
6	GND	RSVDFS	6
7	PWROK	RSVDFS	7
8	RESET#	LDTSTOP#	8
9	+3.3V	+3.3Vaux	9
10	+3.3V	TRST#	10
11	TMS	+3.3V	11
12	TDO	GND	12
13	TDI	SMDAT	13
14	TCK	SMCLK	14
15	GND	GND	15
16	+12V	RSVDFS	16
17	+12V	+12V	17
18	RSVDFS	+12V	18

(a) Host Side socket of HTX connector (measured off mainboard) and compared with: [16]

Pin	A	B	Pin
1	GND	USER2	1
2	REFCLKL	GND	2
3	REFCLKH	GND	3
4	GND	RSVDFS	4
5	GND	RSVDFS	5
6	RSVDFS	GND	6
7	RSVDFS	PWROK	7
8	LDTSTOP#	RESET#	8
9	+3.3Vaux	+3.3V	9
10	TRST#	+3.3V	10
11	+3.3V	TMS	11
12	GND	TDO	12
13	SMDAT	TDI	13
14	SMCLK	TCK	14
15	GND	GND	15
16	RSVDFS	+12V	16
17	+12V	+12V	17
18	+12V	RSVDFS	18

(b) ACE1 gold fingers taken from [4]

It can be seen on Page 12 of the ACE1 schematic[4] that for both the A and the B sides of the small portion of the edge connector (labeled J13-B and J13-D on the Schematic) interface there is a HTX_12V connected to both A99 and B99. This means that even though the edge connector was inverted, the ACE1 board was still able to power up via HTX. An additional effect was to connect the signals: B98-RSVDFS and A100-RSVDFS to HTX_12V due to the fact that pins A98 and A99 combine to become HTX_12V as do pins B99 and B100. The HTX standard requires these lines to be left unconnected and not grounded which is very

fortunate because it saved us from accidentally connecting HTX_12V directly to ground.

We now had to deal with the sizeable problem of trying to correct this issue and invert the small connector on the ACE1 board, while maintaining enough signal integrity for HTX to negotiate the link speed at the minimum frequency of 200MHz (see: Section 4.2.2 for HTX initialization details). The HTX connector form factor specification for HT daughtercards and ATX motherboards lists the following requirements for any daughtercard:

"The HyperTransport connector should be considered a "zero mismatch boundary" as defined in Section 2.3.6.2 of the HyperTransport Interface Design Guide, Rev. 1.07. Motherboard and daughtercard routing should comply with the Interface Design Guide using this assumption. This allows the routing for the motherboard and the daughtercard to be independent and interoperable."

and

"Trace lengths for HyperTransport signals on the motherboard should not exceed 9.25 inches. Trace lengths on the daughtercard should not exceed 2.75 inches. This will ensure that the total trace length will not exceed the maximum allowed 12 inches for 800 MT/sec operation."

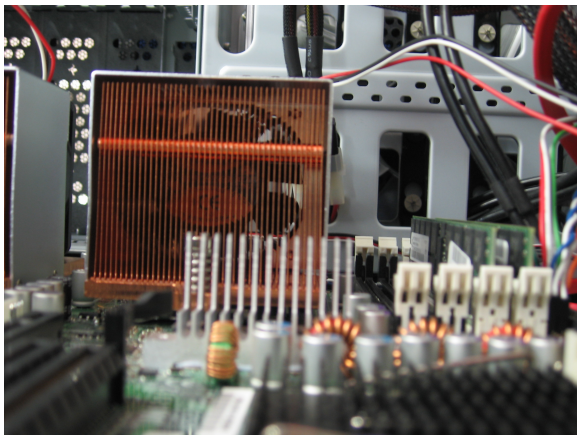
The combination of these two requirements gave us very little total trace length to work with for the converter card and lengths had to be as closely matched as possible. Also the converter would now cause ACE1 to lie parallel to the mainboard and would therefore need to clear certain components on the mainboard, namely the RAM slots and one of the smaller heatsinks that can be seen in Figure 3.12a. The routing space for tracks between the pins of the PCIe type connectors is limited and going around the sides of the connector was not possible as this would have caused significant additional track length, as well as complicating the length matching substantially. Fortunately differential pair signals are usually used because they minimize electromagnetic interference and cross-talk that routing data tracks this close together on a PCB would cause. The schematics and PCB layout can be found in Appendix A and the board was produced by a local PCB manufacturer.

Earlier in this chapter we drew attention to the riser card required for utilizing HTX on the Tyan n3600QX mainboard with images provided in Figure 3.10. As mentioned this riser card was unavailable for us to test and we note that on that card the track lengths for the HTX interface appear very long, certainly longer than the required maximum length presented in the daughtercard standard of 2.75 inches.

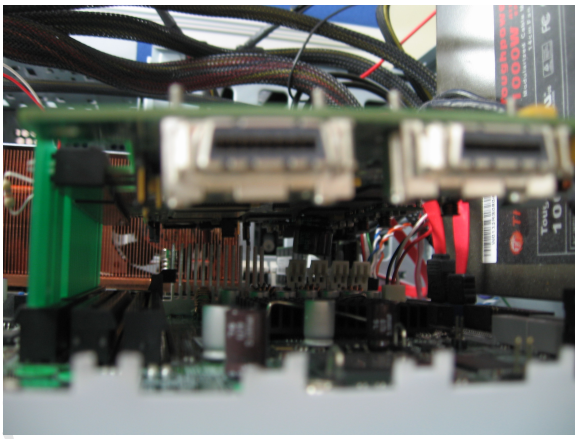
This addition to the overall socketed solution is detrimental to the signal integrity of the HTX interface data signals. This was, perhaps, the moment when a switch to a hardware project developing a re-worked board (ACE1 revision 2, or ACE2) would have been most applicable (See conclusions in Chapter 6). However, the converter did correct: the voltage source problems, the HTX Reference Clock pair: HTX_REFCLK[H:L] and the single ended CMOS initialization signals: PWROK, RESET and LDTSTOP.

We could then move to the next phase of the project: link initialization which we discuss in Section 4.2.2.

A final point with respect to the socketed solution for ACE1 is to clear up any uncertainty surrounding the issue of power reaching the mainboard from ACE1 via the HTX connector. This problem continued despite the converter card switching the signals from the incorrect to the correct side of the edge connector . We give details of the findings in Section 5.1.



(a) Image of Tyan n3600R showing components on the mainboard that the converter card needed to clear while maintaining as short a track length as possible. They are the small silver passive cooler in front of the main CPU cooler and the RAM modules to the right of the main CPU cooler. The end of the HTX connector can be seen (out of focus) to the left of the two PCIe sockets in the bottom left of the image.



(b) Image of Tyan n3600R showing how ACE1 fits into the chassis now lying parallel to the mainboard. The indicator LEDs are obscured because the component layer of ACE1 faces the mainboard. In addition it can be seen that the ACE1 board is not truly parallel to the mainboard. This is due to the PSU12V connector on the far end of the board resting on the RAM modules due to the converter card being slightly too short. Every effort was made to keep it as short as possible while still clearing the mainboard components.

Chapter 4

Gateware design and testing

As presented in Section 3.5, much time was spent dealing with the hosting environment for ACE1. During the delays work was being done trying to prepare the various FPGA cores that would enable access and control of each of the ACE1 interfaces. This included the porting of an HTX end-point gateware core to the v5lx50 and, using an attached DMA application, created a gateware core that allowed us (in simulation until the hosting environment problems were solved) to program the v5lx110t. The final sections of this chapter discuss the gateware collected for supporting the two 10GbE networking interfaces on ACE1.

4.1 Management Gateware

One of the objectives set out in the introduction (Section 1.2) was to allow easy access to the various board resources and interfaces from the host machine. This Section outlines the design that was envisaged, and partially completed, for supporting the ACE1 card once socketed and communicating with the host machine. One of the requirements, outlined in 1.2, was to try and optimize the gateware for lowest possible resource usage, such that as much of the device as possible is left over for user logic. In addition it was desirable that the system be modular so that unused interfaces did not have gateware associated with them, thereby allowing those FPGA resources to be freed up for user code. This system was being constructed during the times that ACE1 was unable to be powered and interfaced with the host machine so we relied on the predicted IO from the UH HTX core to drive the system. Provided along with the UH HTX core is a DMA application discussed in: Section 4.2.4 which pushes packets, stripped of their HT headers, into a BRAM memory bank within the FPGA. Accompanying this memory bank are signals from the *pioengines* (see Section 4.2.4) that signal the start and end of reads and writes.

A block diagram for the Manager gateware is shown in Figure 4.1 and simulations showing the setup and programming of bitstream can be found in Appendix A.

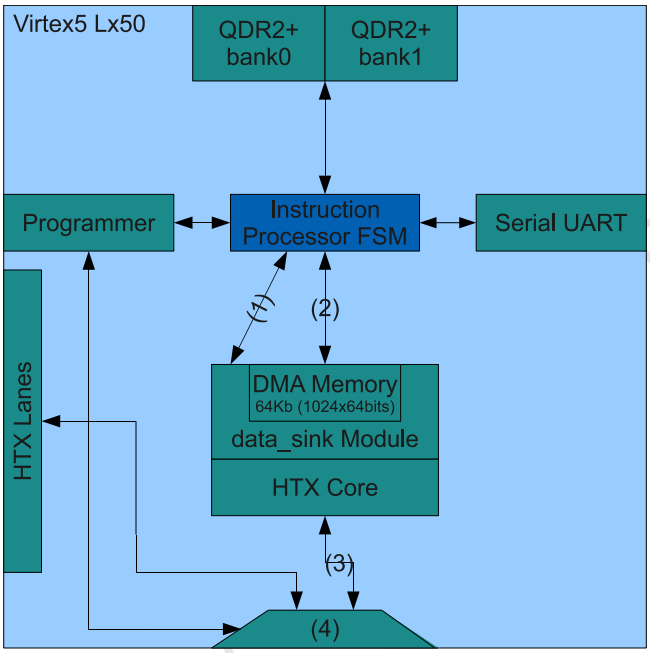


Figure 4.1: Block diagram of the management gateware written for the v5lx50 with the initial features of: programming the v5lx110t and reading and writing QDR2+.

1. DMA status signals:
   dmawrite_start; dmaread_start; interrupt_start; dmawrite_done; dmaread_done; interrupt_done, new_command and cmd

2. BRAM Memory read/write signals:
   enable and address signals: mem_raddr; mem_waddr; mem_wen; mem_out and mem_in

3. HTX signals
   Outlined in Figure 4.4.

4. Mux for switching all HTX lines to the pins associated with the HT link between the v5lx50 and the v5lx110t

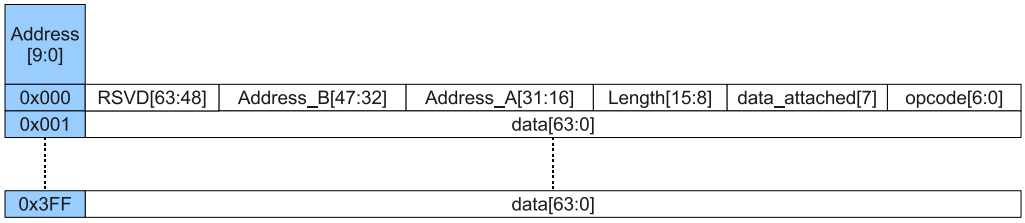


Figure 4.2: Packet structure chosen for Management gateware state machine

Using the data and signals from the *pioengines* as the basis for the Management gateware, a state machine was written to process instructions that the host machine would send via HTX. The memory inferred by the HDL in the UH HTX core is 64Kbits in a 1024 addresses by 64 bit width configuration. As shown in Figure 4.2 we chose the top 64 bits of the memory to specify an instruction and the following 1023 contain any associated data.

The initial, very limited, set of states were dedicated to allowing the host to send a bitstream and instruct that it be used as a configuration bitstream for the v5lx110t. This process was implemented with the intention of extending it substantially once the HTX link was fully tested. As mentioned; the first requirement was designed, built and tested during the time that our host machine was being replaced due to non-conforming connectors so at that stage it was assumed that HTX would provide the host-to-board interface and that the DMA application would be used at least as a starting point for testing.

There was, however, some additional effort that went into proving the viability of this system by reading the bitstream via RS232. This was actually in an attempt to debug the programmer core at a time when it was still reasonable to assume that the HTX problems would be solved. The RS232 FPGA module and host-based python scripts allow pre-crafted packets to be transmitted and de-packetized on the card. These packets are pushed into a BRAM-based memory bank and the same signals as the HTX DMA application would trigger are triggered by the RS232 *de_packetizer* module. This system was useful for showing that the programmer interface worked and would allow us to program the v5lx110t.

Serial RS232 was never an option for a final solution with respect to sending bitstreams to the board. Programming the v5lx110t through this method does not give any significant advantages over JTAG and programming both FPGAs is already supported via JTAG. In addition, care must be taken to avoid any data loss over the RS232 link. In the past corrupt bitstreams would cause damage to FPGAs, however for newer generations of the devices it usually requires a malicious bitstream to be crafted to cause actual chip damage because the FPGAs implement a checksum on the chip before programming [53]. The checksum would reject any damaged bitstreams and therefore require the entire RS232 bitstream to be re-transmitted if it were damaged while being sent. A checksum was added to each of the packets such that only an individual packet needs to be re-transmitted. The request for re-transmission on a damaged packet was not implemented due to the system being abandoned.

The gateware written for this system was tested in simulation and in hardware, however the bitstream was intercepted before being tied to the v5lx110t programming pins. The code and simulations can be found in Appendix A.

4.1.1 HTX chaining problems

There is an additional issue that was discovered while working on the Management gateware. The UH HTX core, that is discussed in Section 4.2 in depth, is what is referred to as a cave device or an end point in an HTX chain, it would be non-trivial to allow the HT chain to extend further using this core. This problem area is expanded on in Section 5.4.

4.2 HTX

HTX is the board-to-host interface. The options for supporting the HTX interface consisted of the following[1]:

1. Altera provide a MegaCore IP library and an HTX IP core exists within their library for use with their products [6]. When the manufacturers distribute these cores they are typically available only as netlists[2] which means that they are not able to be broken down, modified or even inspected without significant effort. Supporting ACE1, a Xilinx Virtex-5 based FPGA co-processor board, using this core would have been very challenging as well as an unacceptable usage of Altera's IP.

2. GDA Technologies Inc. offer a proprietary HyperTransport portfolio of cores including: Cave, Tunnel and Host, Bridge, Switch Port. They have recently upgraded their cores to support the newer HTX3 standard. Attempts were made to establish a cost for using their IP but costs are not listed publicly and would not be discussed until an NDA was signed. The NDA included not discussing the pricing with third parties. Due to the very high NRE costs associated with creating these IP cores it seems likely that they would come at a substantial cost.

3. An open-source core is available from the HTX Center for Excellence based at University of Heidelberg (UH).

[1]Note that the full list of all options are presented for completeness only and to establish the lack of industry adoption for HTX at the time.

[2]The FPGA manufacturers provide a number of intellectual property (IP) cores to users of their devices. Some of these cores require additional licencing fees (such as 10GbE), others do not. The IP cores are often provided by 3rd party companies and simply integrated into the FPGA manufacturers toolchain. This gives companies a chance to provide or sell a specialised core to the whole user base associated with a certain manufacturer as well as giving users options for very efficient interface cores provided by specialists. For a less common interface like HTX there is seldom more than one core available.

4. The final alternative would have been to write a new HTX core.

The University of Heidelberg HTX board was reviewed in chapter 3.1.2 of [5], and their corresponding HTX core, targeted at a Virtex-4 FPGA, was the best choice for a core in terms of openness[3], suitability and time-to-market (in our case time taken to establish board-to-host communications). The steps required to get ACE1 communicating with the host sever are listed:

1. Ensure that the UH HTX core can be built for the target v5lx50 FPGA
2. Ensure that the bitstream is written to the v5lx50 before or during the host's boot sequence
3. Ensure that the core makes it through the initialization sequence
4. Ensure that a driver on the host side can write to, and read from, the co-processor board in some reasonable fashion

Before beginning the discussion on work done for the above list we are going to explain the relevant details for working with the UH HTX core.

4.2.1 Background to University of Heidleberg HTX core

The architecture of the UH HTX link is shown below in 4.3.

There is an older version of the HTX core that only supports 8 bit links in each direction (a total of 16 differential pairs or 32 tracks on the board). This was the core that testing began on, however shortly after work on ACE1 commenced, UH released an updated core which is able to operate at 400MHz DDR with a total link width of 16 bits. ACE1, at the time, was still being laid out so the initial testing began with the 0.9 version of the UH HTX core and then switched when version 1 was released. While we won't discuss the version 0.9 core further due to its lack of suitability in terms of link width supported, we did make the necessary changes to have this design compile[4] for a v5lx50.

Due to the exact clock alignment requirements of HTX each set of 8 lanes is clocked by a dedicated differential clock pair and, in addition, an overall reference clock (HTX_REFCLK) arrives from the host side of the interface. Clocks and lanes for HTX are shown in Figure 4.4. The reference clock remains at 200MHz whereas the lane clocks get ramped up during

[3]Open source gives us the advantage of being able to examine, modify and eventually re-use the core. This feature alone is essential for research projects such as this one. Dealing with any of the manufacturers cores that are available only as a netlist is much more difficult although many of the companies specializing in selling IP cores do give out the full HDL code hence the need for a very strict NDA.

[4]Compilation in this case being the process of creating hardware from HDL code via: Syntax check, Synthesis, Translate, Map and Place & Route

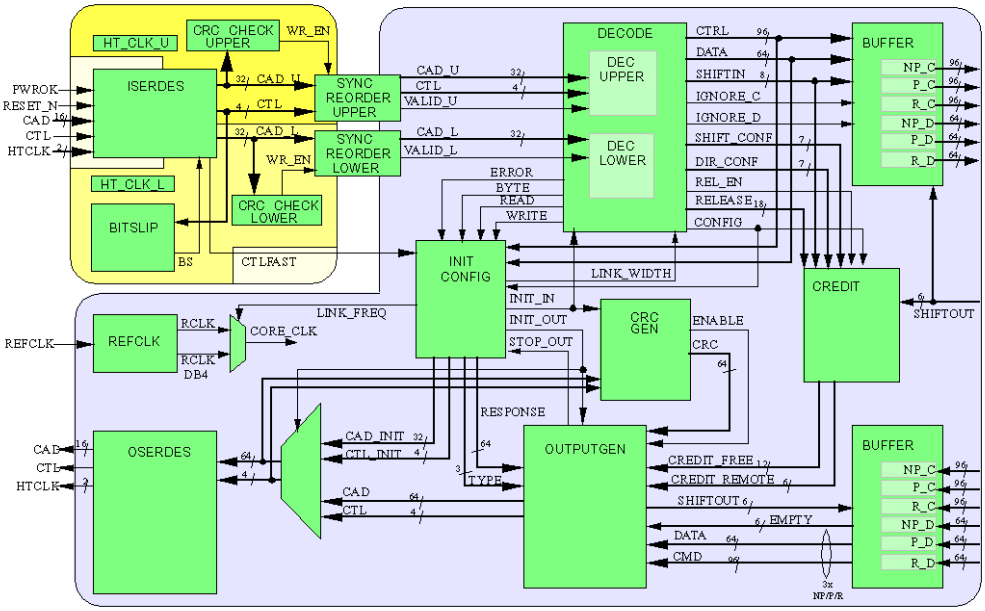


Figure 4.3: Architecture of the HT core supporting 16bit wide links

initialization of the link to the highest negotiable frequency in increments of 200MHz. The Virtex-5 FPGAs on ACE1 are -1 speed grade giving a theoretical maximum frequency of 450MHz and therefore the logic within the device could only support the 200MHz and possibly 400MHz modes of the link. In addition these link clocks are not guaranteed to be stable at all times (they can be manipulated during operation and are only available after the DCM circuits have become locked) therefore the HTX core is clocked off the reference clock at 200MHz and the core utilizes the specialized Serializer-Deserializer (SERDES) hardware blocks within the FPGA to parallelize the inputs and outputs. The serialization is done at a factor of 4 and therefore the maximum speed that the link can attain is 800MHz[5].

As required by the HT specification the UH HTX core supports the base virtual channels: posted requests channel, non-posted requests channel and responses channel [46].

4.2.2 Initialization of HTX

The initialization of an HTX link is broken into a low-level link initialization phase and an I/O chain initialization phase. The low-level initialization of an HTX link defines both a cold reset and a warm reset and relies on the signals: HTX_PWROK, HTX_RESET, HTX_LDTSTOP[6] and the HTX_REFCLK. The basics of the initialization process will be briefly outlined here and a timing diagram is provided in Figure 4.5.

[5]The SERDES blocks in a v5lx50 can serialize or deserialize data by up to a factor of 6 so additional work could be done on this core to get it to communicate with the host at 1200MHz.

[6]HTX_LDTSTOP is only necessary for a warm reset and is not used in the UH HTX core in the initialization module, it is handled separately in the *ldtstop_handler_input* module.

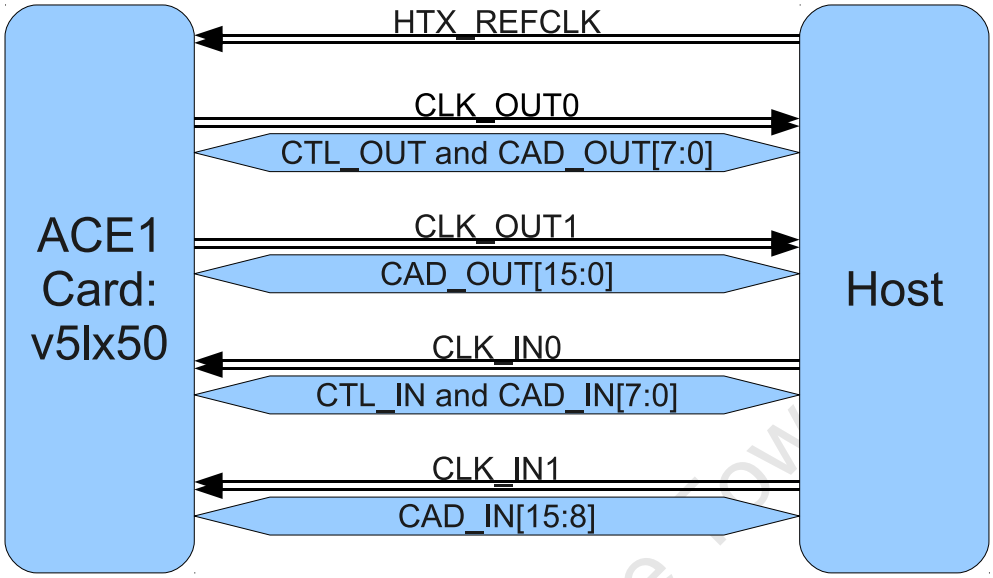


Figure 4.4: Data, control and clock signals used by HT protocol. Excludes power, ground and reserved pins

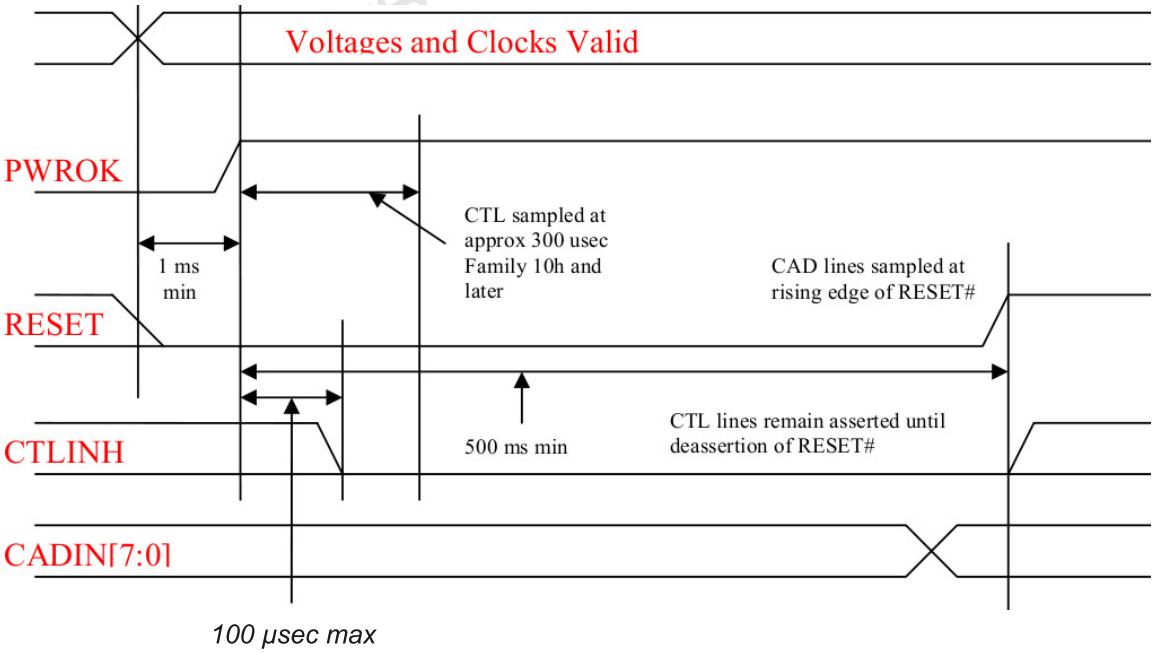


Figure 4.5: HTX initialization timing diagram

HTX_PWROK and HTX_RESET are responsible for signaling that devices are powered and clocks are stable. After HTX_PWROK and HTX_RESET have signaled that clocks are stable and all devices on the link are ready, each side of the link drives the output CAD lines to a logical high to signal its' link width. Each device also samples the incoming CAD lines, thereby reading the link width being transmitted by the other end of the HT chain. Following the link width negotiation the device moves into a warm reset mode whereby the clocks for the data lines (CLK_IN0, CLK_IN1, CLK_OUT0 and CLK_OUT1) are ramped up in increments of 200MHz until one side of the link can no longer achieve clock synchronisation. The data lanes are then clocked at the clock speed of the last successful synchronization.

The UH HTX core handles the low-level initialization phases (cold reset and warm reset) in a module called *ht_init_unit.* In addition to handling the procedure shown in Figure 4.5 the module presents banks of configuration registers (23 sets of 32 bit registers) to the host. These registers are the same as would be found in a PCI interface to allow hardware to identify itself with the bus. This means that, an operating system can access HTX using the same data structures (structs) and creating an HTX driver has the same requirements as creating a PCI driver.

Officially, for HTX versions greater than 1 (and therefore including 1.1 implemented on ACE1), both HTX_RESET and HTX_PWROK should be implemented as open-drain signals by both the mainboard as well as the daughtercard. This allows daughtercards that are hosting devices with slow power-up times, or as in the case of FPGAs devices that require configuration at power-up, to stall the mainboard's initialization phase until the daughtercard is ready. As mentioned in Section 3.3.2 ACE1 takes approximately 6.5 seconds to load the bitstream image from the platform flash into the v5lx50. An initial bitstream must contain the necessary logic for dealing with the link initialization so it is important for the daughtercard to be able to hold HTX_PWROK low and thereby stall the mainboards initialization sequence until the FPGA is configured. We outline our attempts to have ACE1 stall the HTX initialization sequence and the overall HTX non-detect problem in Section 5.2.

A partial solution for getting around the non-detect problems is to pre-power the ACE1 board which could be done once the problems with the 3.3V power supply had been resolved (Section 5.1). However there were still a number of further issues that were encountered while attempting to get the ACE1 board initialized with the host operating system.

The first was that the Tyan n3600R required a BIOS upgrade to clear a POST error: "Problem Fix Description: Fixed an issue where if you would install a 3rd party HTX card the system would hang during POST (Post Code 55)"[7]. A second problem was that the initialization would halt on a BIOS code 87 error unless both devices had been initialized in the correct order from a cold boot. If either device is issued a warm reset then the system freezes during

[7] `http://www.tyan.com/support_download_bios.aspx?model=s.s2912`

boot with a code 87 error. Tyan documentation lists code 87 error as: "Configure Motherboard Devices".

The combination of; the converter card discussed in Section 3.6, pre-powering and programming ACE1, the BIOS upgrade and finally, the modifications to prevent the ACE1 3.3V power supply from rendering the host server un-bootable allowed the ACE1 board to pass HTX initialization off a cold reset.

4.2.3 Upgrading the core for the Virtex-5 Lx50 FPGA

The upgrade for Virtex 5 was straightforward. Although we see some fairly substantial changes to some of the low level hardware when comparing a Virtex-4 FPGA with a Virtex-5 FPGA, the migration documentation from the manufacturer detailed the necessary steps and the toolchain assisted with warnings for any piece of hardware that needed further attention [57, 58]. The Coregen program is provided by Xilinx for assisting in generating any of the Xilinx provided hardware cores. Coregen allows the core specific parameters to be selected or edited and then builds output files for the desired subsystem. The UH HTX core uses only the FIFO Coregen cores in various configurations however, newer versions of the Xilinx tool environment (ISE) did not seem to cleanly upgrade the FIFO cores from project files created using older versions of the tools; they had to be re-created. Coregen stores a configuration file for each generated core in a *.xco* file so creating a new core with the same settings was trivial. In addition, some of the fixed hardware blocks used were changed in Virtex-5, however the toolchain switched them out automatically and issued a warning. Although not technically necessary it was simple to find each item from the warnings list and switch it out in the code. Finally the pin mapping had to be changed to match the assignments on the ACE1 schematic.

4.2.4 UH HTX Application - DMA

As previously mentioned the UH HTX core supports the three base virtual channels and the core has FPGA internal IOs, see Figure 4.3, allowing each of the virtual channels to be written to and read from as appropriate [to the direction of the IO]. In addition to the base UH HTX core UH also provide a *data_sink*[8] module which is a Direct Memory Access (DMA) application which can be attached to the base HTX core. It was decided that this would be a good starting point to work from while testing the interface. The DMA module consists of a number of state machines which sample the incoming packets from the virtual channels, store

[8]The data sink module is the top level module of the DMA application and in turn contains: target memory bank (*memory_I*), block dma read engine (*dmaread_I*), block dma write engine (*dmawrite_I*), PIO read engine (*pioreadengine_I*), PIO write engine (*piowriteengine_I*), command sequencer (*cmd_fsm_I*) and a timer module for determining the speed of the link (*timekeeper_I*)

them in BRAM and produce the appropriate responses to the link, where required. It also includes a timing module used for benchmarking the link.

The constructed bitstream including the *data_sink* module (and its sub-modules) was built using the ISE toolchain.

With all of the infrastructure discussed in Section 3.5 in place and the porting of the core to ACE1's v5lx50 FPGA completed, we were keen to see the test application function and begin working on our own application and drivers. Initial tests for the whole design, including the DMA application, were not successful. The operating system did not detect any new devices in the list of PCI devices. The reasons and possible future solutions for this will be discussed further in Section 5.2 and the further debugging of the HTX core is discussed in Section 5.3.

4.2.5 UH HTX driver

The UH HTX DMA application discussed in the previous section comes with Linux drivers for the host side of the interface. The drivers require Linux kernel headers to be installed before they can be compiled. The operating system chosen for the host server was the most recent long-term-support server version of Ubuntu Linux because; it was most familiar, the kernel headers are trivial to install and the HTX driver compiled on this system without producing errors.

4.3 QDR2+ gateware

The gateware for supporting the QDR2+ modules can be generated using Xilinx Coregen Memory Interface Generator (MIG) . When generating the gateware a top level module can be created, by MIG, that bands together multiple QDR2+ interfaces however, the user code still needs to specify which bank to write to. The instantiation module for the QDR2+ banks was modified to provide a unified address space such that the QDR2+ banks appear as a single continuous address space.

A default QDR2+ interface is provided in Appendix A for using either both QDR2+ banks attached to the v5lx50 or all four QDR2+ banks attached to the v5Lx110t. Xilinx MIG provides a number of alternate configuration options if user code wishes to split up the interfaces or use fewer than the maximum number of RAM banks.

For the v5lx50, that contains the manager gateware at boot time, we instructed the Xilinx MIG to generate a combined gateware interface for both QDR2+ banks such that the manager can read and write to all of the v5lx50 RAM. This means that while the manager module is operational, any user code cannot use the QDR2+ unless it does so by attaching to the manager module. As explained in Section 4.1 the manager gateware reads instructions in

a packetized format and had development on the manager module continued it would have included instructions for reading from and writing to QDR2+.

In testing of the QDR2+ RAM banks we see that the third bank does not achieve calibration at 150MHz while the remaining five banks do. For the v5lx110t we also provide a single gateware module for reading and writing all four of the QDR2+ banks; splitting them off to use fewer than all four and thereby reducing the gateware size can be done using Xilinx MIG. By banding the module interfaces together we limit them all to the same speed, meaning that the three banks that could run at 150MHz are under-clocked. Using this gateware as a template and removing the offending third QDR2+ bank would allow the clock rate to be increased.

We had hoped to be able to provide gateware for managing a proper memory hierarchy allowing the QDR2+ banks to act as a cache for the FPGA co-processor. This would have included a cache-coherent version of the UH HTX core which is available for use from the UH website[9]. The cache coherent version of the UH HTX core requires a coherent HyperTransport license which would need to be obtained from AMD. This design would have relied on our ability to access the main system memory over HTX which by this stage was not an option due to the problems outlined in Section 5.2.

4.4 10GbE Gateware

ACE1 has two IEEE 10GbE-CX4 connectors attached as outlined in Section 3.2.3. Details of the firmware collected to support 10GbE on ACE1 are outlined below. 10GbE is managed by a collection of protocols, that fall into layers, outlined by the OSI stack shown in Figure 4.6. The work done on 10GbE was started towards the end of the project and, while it met with more success than the HTX portions, time constraints limited the development of the support gateware.

4.4.1 Physical Layer

The PHY layers for the 10GbE interfaces are implemented in the FPGA using the Xilinx Rocket-IO hardware as opposed to having an external PHY (often PHY and MAC are done externally and together e.g. Intel's 82599ES dual port 10GbE MAC and PHY [26]). This implementation gives good flexibility and since the PHY layer consists largely of dedicated hardware, primarily the RocketIO transceivers, so implementation of the PHY within the FPGA is not costly in terms of the slice LUTs, slice Registers and BRAM that we would like to leave available to users.

[9] `http://ra.ziti.uni-heidelberg.de/coeht/?page=projects&id=chtcore`

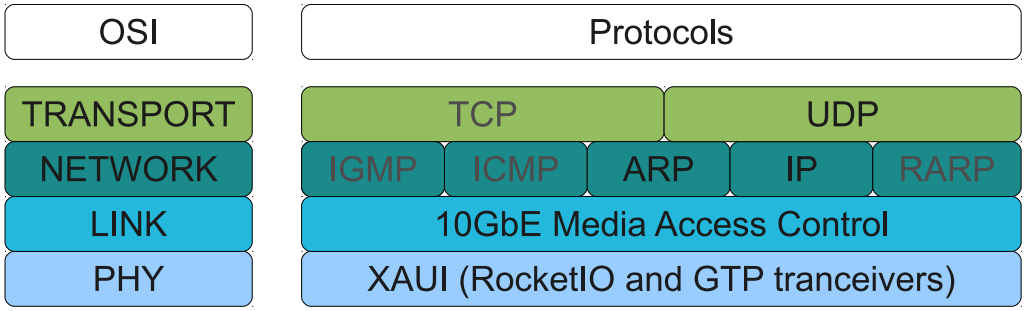


Figure 4.6: 10GbE OSI stack implemented on ACE1. The protocols most likely to be used for moving data between ACE1 boards and other systems on the network are highlighted. In this diagram IP refers to Internet Protocol rather than Intellectual Property.

Xilinx provide their XAUI IP core free of charge. It provides a 10 Gigabit Media Independent Interface (XGMII) to the Rocket-IOs. The only point of interest to users when working with this core on ACE1 is that a pin swap occurred during the layout of the 10GbE interface, which means that the XAUI core requires parameters to be passed to it such that it reverses the polarity of the LVDS channels that were the subject of the pin swap. The swapped pins can be found in the ACE1 schematic [4] on page 25. The parameters passed to XAUI are as follows:

```
1  --VHDL parameter syntax for passing parameters to
2  --the Rocket-IO interface via the rocketio_wrapper module
3  TILE0_TXPOLARITY0 := std_logic := '0'; -- remains unchanged from default
4  TILE0_TXPOLARITY1 := std_logic := '0'; -- remains unchanged from default
5  TILE0_RXPOLARITY0 := std_logic := '0'; -- remains unchanged from default
6  TILE0_RXPOLARITY1 := std_logic := '1'; -- Swapping the polarity for LVDS channel
7  TILE1_TXPOLARITY0 := std_logic := '0'; -- remains unchanged from default
8  TILE1_TXPOLARITY1 := std_logic := '0'; -- remains unchanged from default
9  TILE1_RXPOLARITY0 := std_logic := '0'; -- remains unchanged from default
10 TILE1_RXPOLARITY1 := std_logic := '1'; -- Swapping the polarity for LVDS channel
```

The XAUI core must have its' Rocket-IO tiles constrained to specific GTP sites for each of the two 10GbE interfaces. Along with the site placement for the tiles, the supporting 156.25MHz reference clock must be constrained to correct sites [55, 56]. The following portion of the constraints section of the board support package for ACE1 shows the options for placing the Rocket-IO tiles correctly for each 10GbE interface (or both).

```
# For CX4 on J8 (BOTTOM):
## ----Setting placement for tile0_rocketio_wrapper_i/GTP_DUAL
INST *xaui_block/rocketio_wrapper_i/tile0_rocketio_wrapper_i/
    gtp_dual_i LOC=GTP_DUAL_X0Y2;
## ----Setting placement for tile1_rocketio_wrapper_i/GTP_DUAL
INST *xaui_block/rocketio_wrapper_i/tile0_rocketio_wrapper_i/
    gtp_dual_i LOC=GTP_DUAL_X0Y3;
```

```
## ----Pin locations for accociated reference clock (156.25MHz)
NET "xaui_core1/refclk" PERIOD = 6.4 ns;
NET "xaui_core1/txoutclk*" PERIOD = 6.4ns;
NET "xaui_core1/clk156*" PERIOD = 6.4 ns;
NET "xaui_core1/clk312*" PERIOD = 3.2 ns;
NET REFCLK_n LOC=AF3;
NET REFCLK_P LOC=AF4;

# For CX4 on J5 (TOP):
## ----Setting placement for tile0_rocketio_wrapper_i/GTP_DUAL
INST *xaui_block/rocketio_wrapper_i/tile0_rocketio_wrapper_i/
   gtp_dual_i LOC=GTP_DUAL_X0Y4;
## ----Setting placement for tile1_rocketio_wrapper_i/GTP_DUAL
INST *xaui_block/rocketio_wrapper_i/tile0_rocketio_wrapper_i/
   gtp_dual_i LOC=GTP_DUAL_X0Y5;
## ----Pin locations for accociated reference clock (156.25MHz)
NET "xaui_core2/refclk" PERIOD = 6.4 ns;
NET "xaui_core2/txoutclk*" PERIOD = 6.4ns;
NET "xaui_core2/clk156*" PERIOD = 6.4 ns;
NET "xaui_core2/clk312*" PERIOD = 3.2 ns;
NET REFCLK_n LOC=P3;
NET REFCLK_P LOC=P4;
```

These are the tested constraints and options that the user needs to be aware of when using this core. Additional options such as the pre-emphasis and voltage swing settings can be passed to the XAUI core as parameters.

4.4.2 Link Layer - Media Access Control

A Media Access Controller (MAC) is the next required layer in the Ethernet stack. Research into possible MAC cores was finalized with two options for a MAC on ACE1. The first option is the Xilinx 10GbE MAC although this MAC is only available for evaluation without a proper license. We selected another MAC, developed by the CASPER Project[10] based at the University of California, Berkeley, in conjunction with the MeerKAT[11] project due to ACE1 having been intentionally developed with similar 10GbE interfaces to the MeerKAT hardware. It is available as an open-source MAC, however it is limited and does not support the following features that a full MAC does:

[10] http://casper.berkeley.edu/
[11] http://www.ska.ac.za/meerkat/index.php

1. No configuration
2. No statistics vector
3. No CRC check on receive
4. No interframe minimization
5. Supports only full words or 16 bit words at the input
6. No flow control

The MAC cores both attach to the XAUI interface via XGMII. The Link Layer tends to be the highest layer in the OSI model that is supported in hardware. We will discuss this further in Section 4.4.4 as well as discussing a hardware alternative. The fact that the Network layer and above tend to be handled in software leads to both of the MAC cores having a CPU attachment interface. The Xilinx 10GbE MAC has a PLB46 bus attachment allowing it to be connected to a Microblaze or PowerPC soft-IP CPU or a hard-CPU. The UCB/MeerKAT MAC still uses the older OPB bus to attach to a CPU system. Due to ACE1 not having a CPU on the board we had hoped to allow a CPU on the host server to be responsible for the OSI layers above the Link Layer. With access to the host CPU via HTX unavailable we had to rely on either a soft-CPU or a hardware implementation of the Internet Protocol and TCP/UDP protocols.

For the purposes of testing and verification we ported a Xilinx application MAC Exerciser to be used on the v5lx110t. The MAC Exerciser contains a soft-CPU and we recognise that adding a soft-CPU as a gateware system into the v5lx110t is not ideal from a resource usage perspective. Further details of the system are outlined in Section 4.4.3.

4.4.3 MAC Exerciser

Xilinx provide a MAC exerciser for use on some of their demonstration platforms [59, 34]. We ported this application to the ACE1 for use as an introduction project for users wanting to utilize the 10GbE on ACE1. The MAC exerciser is controlled by a MicroBlaze software application that allows various configurations of the Xilinx MAC to be tested. With some modifications to the design we have provided a sample project for testing each of the 10GbE interfaces on ACE1. The source files, project files and documentation can be found in Appendix A.

When running the MAC exerciser we were able to configure the various PHY interface options, including placing the PHY into loopback mode. This allows a single interface to be tested by instructing the pattern generator to generate and transmit frames which are then captured in the read FIFO.

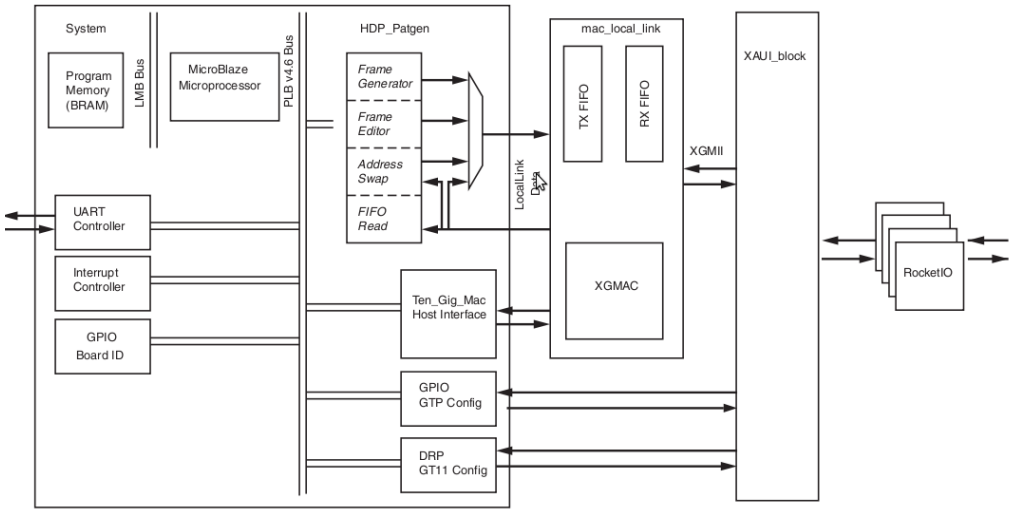


Figure 4.7: Block diagram of Xilinx xapp955 10GbE hardware demonstration platform.

This is a good starting point for a user wanting to utilize 10GbE on ACE1. It includes a soft-CPU as can be seen in Figure 4.7 and basic codes for generating frames using the patern generator hardware. We have already pointed out that running a soft-IP CPU in the v5xl110t is not an ideal solution in terms of resource usage however it does simplify handling the network stack significantly, making it a good starting point for users.

There are a handful of industry products available that support a subset of the features offered by TCP/Internet-Protocol and UDP/Internet-Protocol in hardware which we investigate (in Section 4.4.4) as an alternative to running a soft-IP CPU to support the networking stack.

4.4.4 Network and Transport Layers

The support for the Network and Transport Layer protocols were only briefly investigated in this work due to time constraints. We have already outlined the 10GbE demonstration platform which includes PHY and Link Layer components. The Network and Transport Layer components for this application do not exist, however socket programming is a common task in embedded systems and the software tends to be mature and well documented. The MAC exerciser project provides a starting point for writing the control software for the MAC. Alternatively an embedded OS could be used to control the MAC and PHY and this would allow standard socket programming[12] techniques to be used for both sides of the 10GbE interface. Running an embedded OS on a soft-IP CPU is a common task that is well supported by the FPGA manufacturers toolchain, however it is considered out of scope for this work.

The alternative to handling the Network and Transport Layers with a soft-IP CPU would be to use a hardware-based Internet Protocol stack. There are some interesting projects relating

[12] `http://en.wikipedia.org/wiki/Socket_programming`

to hardware implementations of the UDP and Internet Protocol. TCP is avoided because even simplified versions are too resource intensive to be implemented on an FPGA. Even in the case of UDP, usually not all of the features of the protocol are supported. There are many trade-offs to consider when selecting whether to run a UDP/Internet-Protocol stack in hardware and these can usually be simplified to: "For most systems, it can be stated that no extra functionality than required is desired." - quoted from [32]. There is no easy way to support a general template for implementing a hardware-based UDP/Internet-Protocol stack. This means that for users wanting to use a hardware-based UDP/Internet-Protocol stack on ACE1 they would require in-depth knowledge of the requirements of their system, as well as a good understanding of the various protocols and finally in-depth knowledge of FPGA timing analysis and packing tools.

Many of the hardware UDP/Internet-Protocol stacks are proprietary IP, developed by companies looking to license them. They are likely to be expensive and in our experience the companies do not reveal the price of their products until an NDA is signed, in which there is a clause which disallows discussing (and therefore publishing) the prices. In addition, for hardware-based UDP/Internet-Protocol stacks we find support for speeds up to Gigabit Ethernet only [22].

Our recommendation to users, depending on the requirements of their project, is to implement the Network and Transport Layers of the OSI stack on the soft-IP CPU system already attached to the MAC and PHY in the project that was ported to ACE1, and to avoid a hardware based UDP/Internet-Protocol solution.

An option to further support 10GbE on ACE1 would be to attempt to move the soft-IP CPU to the v5lx50 and use the HTX lanes that exist between the two FPGAs on ACE1 to allow the soft-IP CPU to control the 10GbE MAC. This would free up the resources allocated to the CPU in the v5lx110t and allow better separation between user hardware and gateware as well as allow the system to exist in persistant storage. We discuss this possibility further in Section 5.4.

Chapter 5

Problem areas

In developing gateware for ACE1 a number of significant problems with the board were discovered. As per most hardware designs the first iteration of the design brings to light subtle (and sometimes not so subtle) issues and in many cases multiple iterations are needed achieve a stable system. This chapter starts with the description of the final power related problem with ACE1. We then move to discussing the non-detect problem and some further debugging with ACE1 pre-powered and pre-programmed. The primary complication arose from trying to power up an FPGA-based daughtercard, with a slow configuration time, to communicate with a FSB-like protocol. We conclude by discussing the consequences of the two design decisions that made the largest impact to further ACE1 development; the dual FPGA configuration and the FSB-like board-to-host protocol.

5.1 HTX 3.3V rail problem

One of the problems that was detected while working with ACE1 in a hosted environment was that ACE1 was driving power onto the mainboard when it was socketed in the HTX socket but powered externally. Due to the long initialization time of the v5lx50 it is very useful for testing the HTX initialization process to be able to have the ACE1 board pre-powered from an external source.

Further investigation into this problem led back to the power network shown in [5], specifically section 5.2 and figure 5.7, which shows a switch that determines whether ACE1 gets power from HTX_12V or from the external 12V connector. We note that the schematic does not have distinct symbols for these two power sources and there is no switch to isolate the 3.3V pins and therefore the 3.3V rail can be driven from both the external power supply network on ACE1 as well as the mainboard. This means that when ACE1 is powered externally its power network drives the HTX_3.3V pins and hence provides a current source to the mainboard unintentionally. It is a very undesirable position to be in where a daughtercard with an

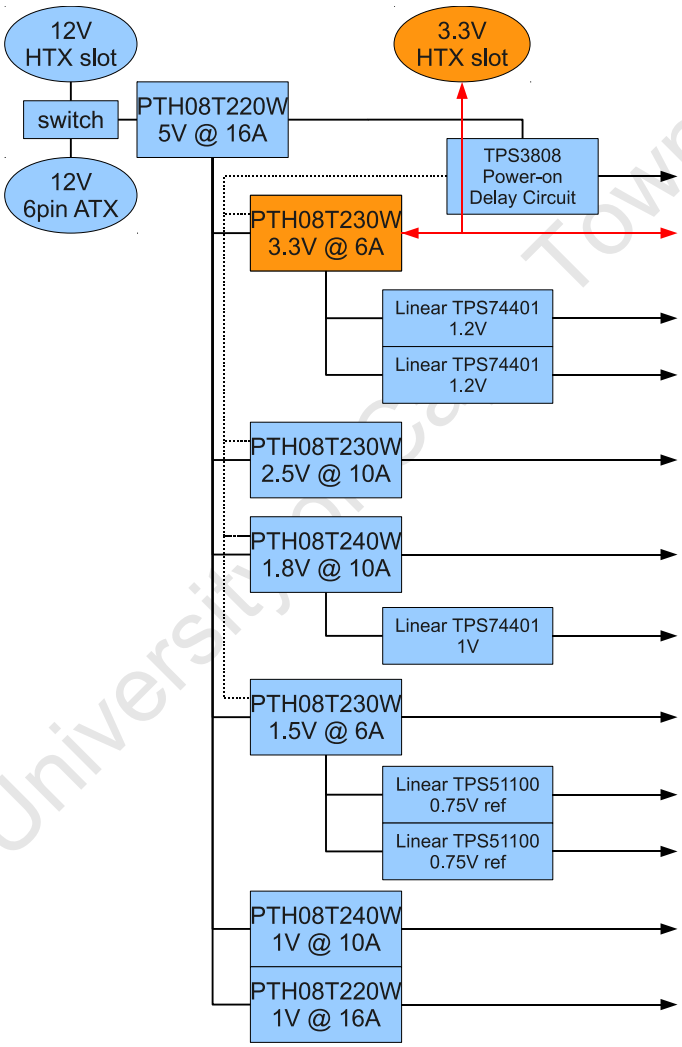


Figure 5.1: Conceptual power network for ACE1 highlighting the problem components.

external power source is leaking power back onto the host system and of course there was concern over damage to the mainboard. This fortunately did not seem to be the case as the host machine functioned correctly without ACE1 connected. It did however prevent ACE1 from being pre-powered from an external power source which was an option for allowing the HTX link initialization to be tested without the 6.5s programming delay.

Due to the ACE1 card requiring only the 12V supplies we removed the 3.3V rails completely by modifying the converter card. This solved the problem of power leaking back to the mainboard and allowed the host server to be booted after ACE1 was powered and programmed (programming via JTAG was then more convenient than using the platform flash).

5.2 HTX non-detect problem

The finalization of work on the HTX interface ended with the host machine operating system successfully detecting the PCI headers[1] but with no read and write access. During the debugging of the HTX interface it was determined that one of the main faults lay in the fact that ACE1 was unable to stall the mainboard power-up sequence to allow the v5lx50 to be programmed with the link negotiation gateware.

The key document for debugging the power supply problems with ACE1 was the HyperTransport ATX/EATX Motherboard/Daughtercard Specification[16] and it includes an appendix titled: "Circuit Design considerations for FPGA-based HTX daughtercards" which lists likely issues caused by the long initialization times of FGPAs on HTX-based daughtercards. We note in Section 3.3.2 that this is the case for the v5lx50 on ACE1 when booting from platform flash. In the case of long initialization times we have; "If an FPGA used as a HyperTransport interface is not initialized in time to respond to the processor during a cold reset, the HTX device may be ignored completely, the system may hang or it may exhibit unpredictable behavior.". It appears that the UH HTX core is being completely ignored by our host server. There are three options presented in the HTX daughtercard specification document outlining the methods for dealing with these delays.

The first attempt at solving the non-detect problem was to force the FPGA side of the interface into an open-drain configuration for both HTX_PWROK and HTX_RESET_N so as to rule out any possibility that the mainboard was skipping past the initialization sequence due to the fact that the FPGA was not yet configured. It is stated by the HT ATX/EATX specification that not all mainboards implement the HTX_PWROK and HTX_RESET_N signals as open-drain signals and it was worth discovering whether the n3600R mainboard conforms to the open-drain requirement. To test this we attempted to hold back the host system boot sequence

[1] Only in the case where the ACE1 board was pre-powered and the FPGA programming time delay problems were circumvented.

by driving the HTX_PWROK and HTX_RESET_N signals from the ACE1 side of the link, however this remained problematic due to the long initialization time. The buffers were inferred using verilog code and the RTL and Floorplanning debugging software tools showed the correct buffers had been created. This test showed that the host system could hold the board back by holding HTX_PWROK low however, the reverse was still not possible because the v5lx50 was not configured in time and pre-powering the ACE1 was not an option due to the power supply problems mentioned in Section 5.1. The only solution to this was to modify the hardware on the ACE1 PCB itself. To avoid cutting up the PCB the changes were implemented on the, suddenly very convenient, converter card where the relevant signals were held to a default low unless driven by the FPGA (only possible after configuration). Despite the HTX_PWROK and HTX_RESET_N being held to ground at all times, after power-up the host continued running through the boot sequence and we were unable to stall the initialization of the HT link. The host server continued to ignore the ACE1 board completely.

The other two options presented in the HTX daughtercard specification document require hardware modification; either:

1. "Design Option 2 – Delaying PWROK" - System designers that have control over the operation of the mainboard could delay the PWROK signal. This is not applicable to ACE1 development however, due to fixes outlined in 5.1, we are able to pre-power ACE1 with similar effects.

2. "Design Option 3 – The "FET" Solution" - a pair of FETs can be attached to the PWROK and #RESET lines to drive them to appropriate values during the first phase of initialization. This would give the FPGA approximately 500 milliseconds of additional time to initialize, however ACE1 is unable to acheive this using Master(or Slave)-Serial programming modes. A further, more complicated version would be to attach FETs to each of the data lines effectively taking over the initialization of the link and then passing control back to the FPGA once programmed. The HTX daughtercard document outlines negatives of component count, routing difficulties and possible signal integrity problems.

Without significant hardware changes to the ACE1 board we are unable to power-up our co-processor board alongside the host server and have it recognised.

5.3 Debugging HTX with ACE1 pre-powered

Due to modifications to the 3.3V power supply it was possible to pre-power the ACE1 board from an external bench power supply completely removed from the host server. This allowed us to continue testing the HTX interface as a proof of concept for the link. Pre-powering the

ACE1 board allowed us to program an HTX compatible bitstream on the v5lx50 such that the host server saw no FPGA programming delays and thereby avoided the problem outlined in Section 5.2. As mentioned in Sections 4.2.4 and 4.2.5 we already had pre-compiled bitstreams updated for Virtex 5 FPGAs and a driver compiled for the host server. With this system in place we were able to have ACE1 successfully identified by the host server via the PCI compatible header registers. We note that the link initialization signals, shown in Figure 4.5, comprise of low speed CMOS tracks.

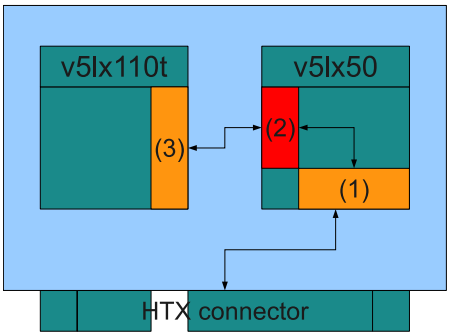
While the board was identified by the host server OS we were still unable to do read and write transfers to the card and further testing revealed that the HTX interface was being negotiated with a link width of zero. All four of the lane clocks were re-checked and appeared stable at 200MHz. We were unable to check, during link initialization, if the clocks ramped up to 400MHz however they should still return to 200MHz if 400MHz is unachievable. The most likely explanation for the initialization failing to negotiate a suitable link width is the addition of the converter card and associated poor signal integrity on the high speed tracks. Coupled with the fact that ACE1 was unusable as a host-based co-processor due to the power-up delay related problems, it was decided not to pursue this issue further.

5.4 HTX chaining problems

There is an additional issue with HTX on ACE1 that was discovered while working on the management gateware core. HT can support direct access to both of the FPGAs on ACE1 if each is enabled with the correct gateware. Each would appear as an independent device in the HT chain.

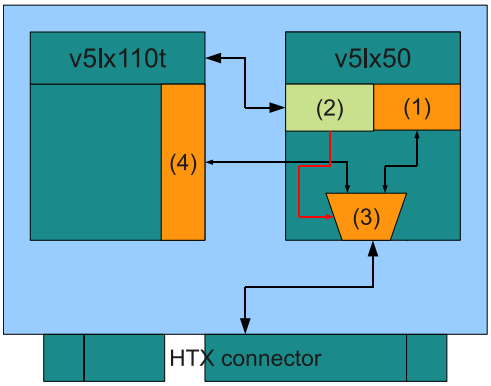
The problem arises from the fact that the v5lx50 can take an image at boot time but it is required to program the v5lx110t (unless we resort to JTAG). The HT1.1 protocol can handle hot plugging a device and this would then be a requirement if support for host-based access to both FGPAs were desired. To support this dual HT node configuration each FPGA device in the HT chain would require the appropriate gateware. The UH HTX core, discussed in Section 4.2, provides us only with the cave device and not the tunnel device[2] which would be required on the v5lx50 for extending the chain through to the v5lx110t. Figure 5.2a shows how the card would be set up and the section shown in red shows the requirements that are unsupported by the UH HTX cave. In addition the UH HTX core is large as far as gateware is concerned; if the DMA application is included for the v5lx50 we see a slice utilization of 15% (slice registers of 15% and slice LUTs at 14%). Further adding to it by converting it into a HT tunnel would consume additional resources from the v5lx50.

[2]An HTX gateware core would be referred to as a tunnel device if it is more than just an end point to an HT chain, it must allow the chain to propagate further as well as supporting access to the device on which it is located.



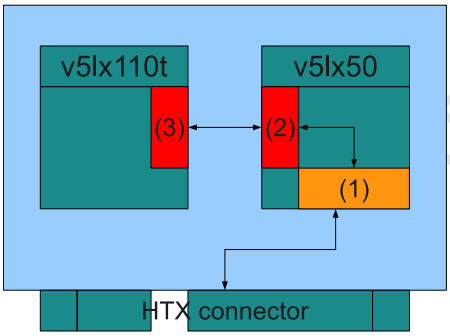
(1)UM HTX cave core supporting host based reads and writes to v5lx50
(2)Additional gateware required to extend the HT chain to the v5lx110t, not supported by current UM HTX core
(3)Second UM HTX cave core supporting host based reads and writes to v5lx110t

(a) Figure showing required HTX gateware to support a full HT chain in both FGPAs.



(1)An initial UM HTX cave core supporting host based reads and writes used during boot up.
(2)Gateware writes an HTX enabled bitstream to the v5lx110t then signals a switch to all of the HTX_LINKCLK,HTX_CAD and HTX_CTL signals.
(3)Multiplexer for switching signals to the newly programmed v5lx110t.
(4)UM HTX core allowing access to the v5lx110t.

(b) A graphical description of the second option for gaining access to the v5lx110t on ACE1 from the host server via HTX.



(1)UM HTX cave core supporting host based reads and writes to v5lx50
(2) & (3) Custom interface using HT differential pairs but not following HT protocols and consequently unable to be attached to the HT chain. Possibly a bus like PLB46 could be operated across the lanes.

(c) Gateware description for allowing HTX access only to the v5lx50 and passing further data via a custom interface.

A second option, shown in Figure 5.2b, is to have the v5lx50 power up with and establish a connection with the HTX chain. It then programs an HTX enabled bitstream from the host server to the v5lx110t after which it switches the HTX lane signals to the v5lx110t. This would be the equivalent of hot-un-plugging followed by hot-plugging a new device onto the HT chain. It is possible that the gateware could be written such that the rest of the system would be unaware of the switch however, without the initial working link, we are unable to confirm this possibility. It seems likely that once the v5lx110t is on the HT chain, using the method outlined as the second option that no further changes would be necessary however ideally the process should be reversible such that we can switch back to communicating with the v5lx50, fetching another bitstream from the host and returning communication to the v5lx110t.

The final option is to attempt to use the HT lanes that exist between the v5lx110t and the v5lx50 as a bridge for the buses supported by the Xilinx EDK toolchain. If this sizable project could be accomplished then the v5lx50 could be dedicated to run a significant soft-IP CPU (possibilities include a Cortex-ARM9 or a PowerPC) and ACE1 could continue being developed as a non-host-based processing board as shown in Figure 5.2. This solution would demonstrate the versatility of the FPGA devices and allow ACE1 to become a viable stand-alone processing platform.

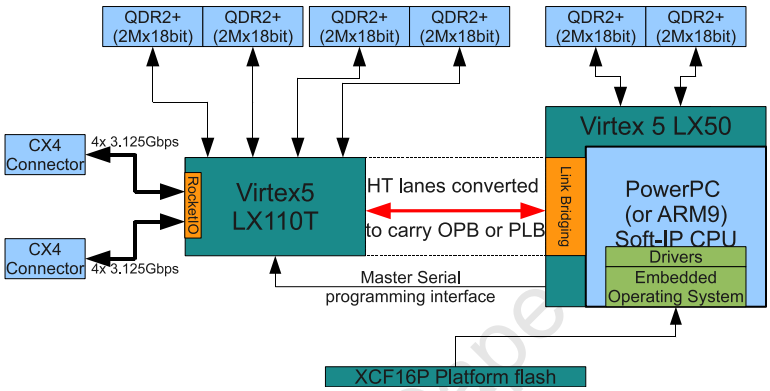


Figure 5.2: Overview of ACE1 as a standalone processing board.

The most significant complication is the fact that both the supported Xilinx options: the older OPB bus and the more recent IBM PLB46 bus, have higher signal counts than the version of HT on ACE1 (even in the case that the HT differential pairs could be split and used independently).

5.5 Serial programming interfaces

The serial RS232 Section (3.2.4) already outlines the problem with RS232 on ACE1. The serial RS232 consists of two sets of transmit (Tx) and receive (Rx) channels between the v5lx50 and a MAX3388E serial PHY device which, in turn, connect to the pin headers on the PCB as shown in Figure 5.3. Allowing RS232 access to the v5lx110t requires relaying these signals through the v5lx50. While hardware for handling this is trivial[3] it does require the use of at least two sets of differential pairs that are part of the HTX link between the v5lx50 and the v5lx110t which effectively disables this interface. While it would be possible to allow the HT protocol to negotiate a link with fewer lanes, thus freeing up lanes that the RS232 serial could utilize, this would have to be done for the HT gateware cores on both FPGAs due to the HTX chaining problem mentioned in Section 5.4, of which our solution, although theoretical, is to hot-plug the v5lx110t device into the chain followed by a communication handoff. In

[3]We provide this as part of the gateware provided to users for ACE1 and it can be found in Appendix A

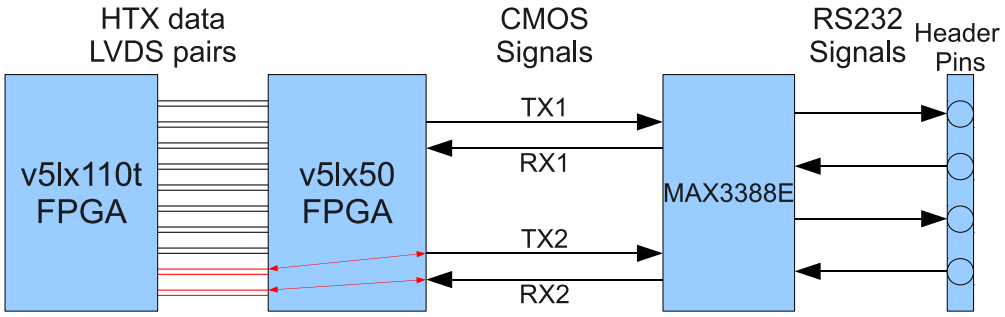


Figure 5.3: RS232 infrastructure on ACE1 showing that bridging RS232 to the v5lx110t consumes a pair of differential tracks.

addition, a fifty percent performance penalty to the HTX link just to give serial access to the v5lx110t is unlikely to be worthwhile. Future versions of ACE1 should correct this issue by either giving direct access to the second FPGA (if there is one) or by removing RS232 altogether and relying on JTAG for debugging.

5.6 Financial Objectives

The initial work on the ACE1 daughtercard provided very few details regarding the final price of the card and is accurate in saying that a comparison between an educational hardware project like ACE1 and a commercial one is impossible. We have a situation where many of the devices are donated, as well as relying on academic versions of the various CAD software programs used in the design. Contrasted with the academic savings, is the significantly greater cost incurred per unit for small orders in the electronics industry. Despite the lack of a suitable comparison for the manufacturing and component costs of the project, we feel that there is merit in outlining the consequences of the initial design decisions on the final cost of the complete accelerator system.

The options for HTX compatible mainboards are listed by the HyperTransport Consortium and the list includes approximately twenty mainboards. Our experience with the Tyan n3600QX and n3600R boards was that the HTX socket tends to be connected to the second CPU socket on the mainboard [44, 45]. This means that the second CPU socket has to be populated for HTX to function and the mainboards have to be populated in numerical order. Ordering a second Opteron 8350 pushed the price of our host machine up significantly. A summary spreadsheet of parts and costs for the ACE1 host machine can be found in Appendix A.

Approximately ZAR 36,600 was spent on the host server for ACE1 with almost half of that cost going into the purchase of the Opteron CPUs. Due to both sockets being populated the host server has 8 CPU cores and a significant amount of system memory, plus peripherals

and HTX support. The idea behind co-processing is to offload the computationally intense portions of the code to the co-processor, theoretically and ideally, leaving the 8 CPU cores idle except for minimal control code. The high CPU count means that the host system is capable of significant computational performance on its own and leaves the need for a co-processor questionable. Contrast this to the Nallatech PCIe-180 or a PCIe version of ACE1 which could be hosted in just about any computer system manufactured since January 2007 when PCIe version 2 was released. An example is provided in the costing spreadsheet for a Dell PCIe host machine with similar memory for approximately ZAR 8,300. The size, complexity and cost of an HTX host server are an unfortunate side effect of the original design decision.

Chapter 6

Concluding Remarks

Overall the work on ACE1 was unsuccessful in terms of the original goals, the first of which was to establish a suitable link between the host server and daughtercard. A considerable amount of time and effort went into solving the problems with HTX as they were discovered and the lack of a working host-based interface caused problems in each of the subsequent objectives. Goals such as attempting to optimise gateware for low resource usage and providing an ecosystem including drivers and a library of bitstreams for supporting various modes of operation were abandoned.

The fundamental design issues with the ACE1 board are summarised below:

1. Difficulties in procuring HTX compatible parts and the consequently high cost of the hosting environment.
2. The HTX 3.3V rail problem - ACE1 leaking power back to the mainboard.
3. The HTX non-detect problem - The added converter card caused the HTX to negotiate a link width of zero due to poor signal integrity.
4. The programming time problem - ACE1 being unable to hold back the boot procedure on the host server to allow time for the v5lx50 FPGA to be configured.
5. The HTX chaining problem - The lack of an HTX tunnel core would have made the task of accessing the v5lx110t from the host server a significant challenge, assuming that point three of this list had been overcome.
6. The serial debugging interface only being available if using two HTX differential pairs - Relatively minor issue due to JTAG debugging still being available.

While this was disheartening, to say the least, a proof of concept had to be completed for ACE1 even if it resulted in a negative conclusion. Inexperience caused me to miss the point

in the project where switching to one of the two options presented below would have been the better decision.

While ACE1 was in the power-up testing phase we noted the appearance of the Nallatech PCIe-180 in the marketplace. This vindicated all except one of our original design decisions. With the exception of HTX, the ACE1 had all the features offered by the PCIe-180 and an additional 10GbE interface and finally an advantage of approximately one year (sadly spent on debugging HTX) over the PCIe-180 board.

6.1 Future work: ACE1 stand-alone accelerator

The final work in this project, the gateware supporting 10GbE, which was done towards the end of the project and with heavy time constraints, showed the most promise. Our recommendation for ACE1 is to abandon any attempts at communication with the host over HTX and focus on the 10GbE interfaces exclusively. Powered externally, and with programming options of JTAG and platform flash, the ACE1 can still be used for accelerating many networking-based applications. We have discussed the HTX chaining problems in Chapter 5 and have noted that the interface signals between the two FPGAs could support an alternate bus system (hopefully PLB46 via some bridging gateware). In addition the v5lx50 is suitable for running a significant soft-IP CPU and embedded Operating System. Future work on ACE1 should be focused in the direction of a standalone system with soft-IP CPU and embedded Operating System exclusively controlling the 10GbE interfaces while the FPGA (now a co-processor to the soft-IP CPU) handles packet-related tasks.

6.2 Future work: ACE2

The next step for continuing work on a co-processor board for a host-based CPU system is to re-work the ACE1 board. Without the power and signal integrity issues that the converter board introduced into the system we recognise that the latency advantages of the HTX interface when compared to PCIe are significant to a co-processor system. However the time spent on finding compliant parts, and consequently the increased cost and time delays have caused us to recommend a switch to PCIe. We have discussed the hard-IP blocks, provided by the manufacturers, in the latest generations of FPGAs that offer further support for PCIe.

Incorporating an improved programming model into the second revision of the project would be a sizable improvement despite the cost of the additional pins. There are a number of options for allowing shorter configuration times available for Xilinx FPGAs. The original Master-Serial programming interface should be upgraded.

The final recommendation is a thorough investigation into the possibility of utilizing partial reconfiguration to eliminate the need for a second controlling device (FPGA or CPLD) on the daughtercard. As discussed in Section 2.11 we note that the v5lx50 is not an ideal device for use as glue-logic only. When designed it was intended the manufacturer tools have begun pushing the partial reconfiguration feature in recent versions of the FPGA toolchains. The inclusion of hardware and firmware blocks such as: hard-IP PCIe and the XAUI soft-IP wrapped around hard Rocket-IO transceivers means that a co-processor board could be simplified down to a single and much larger FPGA using partial reconfiguration. This would shift the interface between user synthesized-hardware and interface gateware into the FPGA fabric and therefore it would be dealt with in a re-programmable environment which would vastly decrease the risks that an error on the PCB would cripple the system.

Appendix A

Design-files and Documentation

Design-files, gateware projects and documentation can be found on the attached CD.

Bibliography

[1] World salaries. URL www.worldsalaries.org.

[2] Microprocessor design. Online. URL http://en.wikibooks.org/wiki/Microprocessor_Design/Microprocessors.

[3] Ross D. Adamson. Novel methods for large molecules in quantum chemistry. Online, 1999. URL http://www-theor.ch.cam.ac.uk/people/ross/thesis/node97.html.

[4] Michael Aitken. *ACE1 Schematic*.

[5] Michael Aitken. A reconfigurable accelerator card for high performance computing. Master of science in electrical engineering, University of Cape Town, 2008.

[6] Altera. *HyperTransport MegaCore Function User Guide*. Altera, .

[7] Altera. Intel atom industrial reference platform, . URL http://www.altera.com/end-markets/industrial/io-hub/intel/ind-msc-platform.html.

[8] Altera. Pcie hard ip. . URL http://www.altera.com/technology/high_speed/protocols/pcie-hard-ip/pro-hard-ip.html.

[9] AMD. The industry-changing impact of accelerated computing.

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

[11] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html.

[12] Krste Asanovic, Ras Bodik, James Demmel, Tony Keaveny, Kurt Keutzer, John D. Kubiatowicz, Edward A. Lee, Nelson Morgan, George Necula, David A. Patterson, Koushik Sen,

John Wawrzynek, David Wessel, and Katherine A. Yelick. The parallel computing laboratory at u.c. berkeley: A research agenda based on the berkeley view. Technical Report UCB/EECS-2008-23, EECS Department, University of California, Berkeley, Mar 2008. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-23.html.

[13] Peter Bakkum and Kevin Skadron. Accelerating sql database operations on a gpu with cuda. *ACM*, 2010.

[14] C. Belady and C. Malone. Data center power projections to 2014. *Thermal and Thermomechanical Phenomena in Electronics Systems, 2006. ITHERM '06. The Tenth Intersociety Conference on*, pages 439 –444, may. 2006. ISSN 1087-9870. doi: 10.1109/ITHERM.2006.1645376.

[15] HyperTransport Consortium. Hypertransport products page. online, . URL http://www.hypertransport.org/default.cfm?page=ProductsProductsByType.

[16] HyperTransport Technology Consortium. *HyperTransport ATX/EATX Motherboard/-Daughtercard Specification Including Errata Revision F HTX Connector and Form Factor Specification for HyperTransport Daughtercards and ATX/EATX Motherboards*. HyperTransport Technology Consortium, .

[17] HyperTransport Technology Consortium. *HyperTransportTM I/O Link Specification Revision 1.10*. HyperTransport Technology Consortium, .

[18] Khaled Ibrahim Alex Kaiser Alice Koniges Kamesh Madduri John Shalf Erich Strohmaier Samuel Williams David Bailey, James Demmel. A testbed of parallel kernels for computer science research.

[19] Eskom. Electricity price increases. URL http://www.eskom.co.za/live/content.php?Item_ID=937.

[20] Ian Foster. Parallel algorithm examples. Online, 1995. URL http://www.mcs.anl.gov/~itf/dbpp/text/node10.html.

[21] John L. Gustafson. Reevaluating amdahl's law. *Communications of the ACM*, 31:532–533, 1988.

[22] F.L. Herrmann, G. Perin, J.P.J. de Freitas, R. Bertagnolli, and J.B. dos Santos Martins. A gigabit udp/ip network stack in fpga. *Electronics, Circuits, and Systems, 2009. ICECS 2009. 16th IEEE International Conference on*, pages 836 –839, dec. 2009. doi: 10.1109/ICECS.2009.5410757.

[23] Brian Holden. Latency comparison between hypertransport and pci-express in communications systems. White Paper.

[24] C.-H. Hsu, W.-C. Feng, and J.S. Archuleta. Towards efficient supercomputing: a quest for the right metric. *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, page 8 pp., apr. 2005. doi: 10.1109/IPDPS.2005.440.

[25] Intel. IntelÂ® ep80579 software for security applications on intelÂ® quickassist technology. Technical report, .

[26] Intel. Intel ethernet controllers and phys, .

[27] IPBlaze. Overview of ipblaze 10g nic in xilinx virtex 5.

[28] S. Kamil, J. Shalf, and E. Strohmaier. Power efficiency in high performance computing. *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1 –8, apr. 2008. ISSN 1530-2075. doi: 10.1109/IPDPS.2008.4536223.

[29] Yousef Khalilollahi. What platform asics are and when to use them. Online, 2005. URL `http://www.design-reuse.com/articles/9443/what-platform-asics-are-and-when-to-use-them.html`.

[30] Bryan Christopher Catanzaro Joseph James Gebis Parry Husbands Kurt Keutzer David A. Patterson William Lester Plishker John Shalf Samuel Webb Williams Katherine A. Yelick Krste Asanovic, Ras Bodik. The landscape of parallel computing research: A view from berkeley. 2006.

[31] I. Kuon and J. Rose. Measuring the gap between fpgas and asics. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(2):203 –215, feb. 2007. ISSN 0278-0070. doi: 10.1109/TCAD.2006.884574.

[32] A. Lofgren, L. Lodesten, S. Sjoholm, and H. Hansson. An analysis of fpga-based udp/ip stack parallelism for embedded ethernet connectivity. *NORCHIP Conference, 2005. 23rd*, pages 94 – 97, nov. 2005. doi: 10.1109/NORCHP.2005.1596997.

[33] Christian Peter Manko. Project report 10 gbe. .

[34] Christian Peter Manko. 10 gigabit implementation in a fpga. Master's thesis, Max-Planck-Institute for Radio Astronomy, .

[35] Rick Merritt. Intel packs atom into tablets, set-tops, fpgas. Online. URL `http://www.eetimes.com/electronics-news/4207694/Intel-packs-Atom-into-tablets--set-tops--FPGAs`.

[36] Gordon Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, 1965.

[37] Nvidia. High performance computing (hpc) - supercomputing with nvidia tesla. Online. URL `http://www.nvidia.com/object/tesla_computing_solutions.html`.

[38] R.M Ramanathan. Extending the worlds most popular processor architecture.

[39] Jeremy W. Sheaffer Kevin Skadron Shuai Che, Jie Li and John Lach. Accelerating compute-intensive applications with gpus and fpgas.

[40] Gregory D. Peterson Shuang Gao. Gpu accelerated scalable parallel random number generators.

[41] Theoretical and Computational Biophysics Group. Gpu acceleration of molecular modeling applications. Online. URL `http://www.ks.uiuc.edu/Research/gpu/`.

[42] John Timmer. Datacenter energy costs outpacing hardware prices. online. URL `http://arstechnica.com/business/news/2009/10/datacenter-energy-costs-outpacing-hardware-prices.ars`.

[43] Tyan. *TYAN Compatibility Test Plan/Report*. Tyan, . Document can be found in attached files Appendix.

[44] Tyan. *Thunder n3600QX Manual*. Tyan, .

[45] Tyan. *Thunder n3600R Manual*. Tyan, .

[46] unknown. Hypertransport cpu technology. URL `http://cpu-hypertransport.blogspot.com/2007/06/io-streams-in-hypertransport-technology.html`.

[47] wikipedia. http://en.wikipedia.org/wiki/coprocessor. Technical report, .

[48] wikipedia. Frontside bus pros and cons. Online, . URL `http://en.wikipedia.org/wiki/Front-side_bus`.

[49] wikipedia. Moore's law. Online, . URL `http://en.wikipedia.org/wiki/Moore's_law`.

[50] Xilinx. *LogiCORE 10-Gigabit Ethernet MAC v8.0 User Guide*. Xilinx, .

[51] Xilinx. Xilinx unveils arm-based processing architecture for delivering unrivaled levels of performance in embedded systems. . URL `http://press.xilinx.com/phoenix.zhtml?c=212763&p=irol-newsArticle&ID=1418796`.

[52] Xilinx. *LogiCORE IP XAUI v9.2 User Guide*. Xilinx, .

[53] Xilinx. *Virtex-5 FPGA Configuration User Guide*. Xilinx, .

[54] Xilinx. *Platform Flash In-System Programmable Configuration PROMs*. Xilinx, .

[55] Xilinx. *LogiCORE IP XAUI v9.2 User Guide*. Xilinx, .

[56] Xilinx. *RocketIO Transceiver User Guide*. Xilinx, .

[57] Xilinx. *Virtex-5 Libraries Guide for HDL Designs*. Xilinx, .

[58] Xilinx. *FIFO Generator Migration Guide*. Xilinx, .

[59] Xilinx. *10-Gigabit Ethernet Hardware Demonstration Platform*. Xilinx, .

[60] Xilinx. Pcie hard ip. . URL `http://www.xilinx.com/products/ipcenter/V6_PCI_Express_Block.htm`.