The Design and Implementation of a Carrier Card for the Karoo Array Telescope

Prepared by
David George

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

May 23, 2008

Declaration

I declare that this dissertation is my own unaided work. It is being submitted for the degree of Masters in Electrical Engineering at the University of Cape Town. It has not been submitted before for any degree or examination at any other university.

Signed by candidate

.....

Signature of Author

May 23, 2008

Acknowledgements

Firstly, I would like to thank the whole KAT DSP team for all their help. Specifically: Alan Langman for his guidance, Francois for his assistance with hardware and Andrew for his ideas about the gateware. Thanks also to my supervisor, Professor Inggs, for his advice. I'd like to thank my mother for, willingly, performing the loathsome task of editing this document. Finally, I would like to thank my girlfriend, Lisa, for her encouragement throughout the project.

Contents

List of Figures

List of Tables

Nomenclature

FPGA	Field Programmable Gate Array. A semiconductor device capable of synthesizing complex digital logic designs
HDL	Hardware Description Language. A code for describing digital logic designs
Verilog	An HDL code
VHDL	VLSI HDL code
VLSI	Very Large Scale Integration
Gateware	A digital design to be implemented on an FPGA.
Bitfile	Compiled gateware used to configure an FPGA
LVDS	Low-Voltage Differential Swing. A high-speed, differential, digital logic family
LVTTL	Low-Voltage Transistor-Transistor Logic. A single-ended logic family
LVCMOS	Low-Voltage Ceramic Metal-Oxide Semiconductor. A single-ended logic family
ADC	Analogue to Digital Converter
DDR	Double Data Rate. I/O system in which data is latched on both positive and negative edges of a clock
SDRAM	Synchronous Dynamic Random Access Memory. A high-volume solid state computer memory
Compact PXI	A backplane architecture based on Compact-PCI, which includes support for timing and synchronization
BER	Bit-Error Rate or Bit-Error Ratio

1 Introduction

The Karoo Array Telescope [KAT] is a South African project that is attempting to build a world-class radio telescope in the Northern Cape. The first prototype phase of the project was called the eXperimental Development Model or XDM. This MSc project involves the development of a carrier card that was planned to be used for XDM. The card, called the XDM Carrier Card, or XCC, was designed to be used as part of a modular Digital Signal Processing [DSP] architecture.

The XCC's external connections are summarized in the context diagram in Figure 1. The card was designed to support two daughter cards, high-speed data network connections, a control network connection and power and synchronization signals from a backplane. The modular architecture was chosen to allow the boards to operate in different modes, depending on the application, for example, as digital receivers or data processors. The primary function of the carrier card was to perform communication and control tasks.

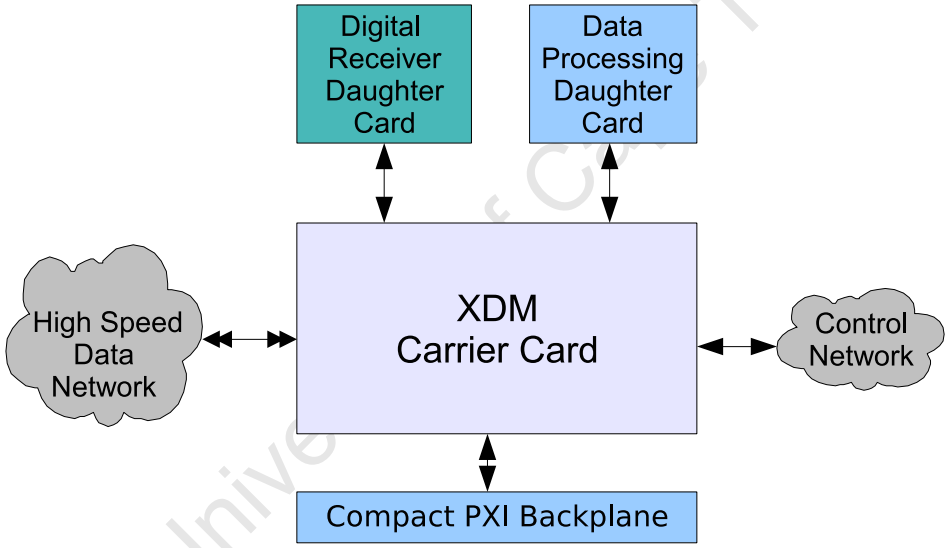


Figure 1: XDM Carrier Card context diagram

Typically, radio astronomy applications require substantial DSP hardware to sample, process and transfer incoming data. The demands on DSP performance are growing rapidly, as new instruments incorporate more dishes and seek improved performance. In the past, these instruments were designed for very narrow applications and took anything between three and ten years to develop. However, the fast growth of the processing power of electronics often rendered these designs obsolete even before they were deployed.

In the last decade, Field Programmable Gate Arrays [FPGAs] have provided an agile means of performing DSP. They facilitate rapid development of digital systems and lend themselves to relatively

simple hardware as they increasingly allow more tasks to be performed on a single Integrated Circuit [IC]. These features allow for frequent updates of hardware, thus taking advantage of the higher logic volumes and clock speeds that new generation FPGAs provide. For these reasons, FPGAs have been widely adopted for use in radio astronomy projects, such as the Allen Telescope Array and in KAT itself. Thus, FPGAs form a central theme in the design of the XCC.

1.1 Project Objectives

The objective of this project was to deliver hardware to be used as part of the Digital Back-End[DBE] of the Karoo Array Telescope for XDM. The hardware was required to include the functionality to satisfy XDM requirements, as well as to provide a test bed for design concepts for later KAT design iterations. Ultimately, the development of the board would serve as a risk reduction exercise in preparation for the development of the more complicated final KAT.

The specific objectives of this project were to:

- Analyze requirements and establish specifications for the XCC
- Design and develop the hardware for the XCC
- Implement gateware for both low-level testing and XDM functionality
- Develop test software for system verification
- Analyse hardware performance and verify board capabilities

1.2 Project Outline

This section briefly provides an overview of the document on a chapter-by-chapter basis.

1.2.1 Chapter 2: User Requirements

XDM, the first prototype of the Karoo Array Telescope, consisted of a single dish system with a bandwidth of 1.414 to 1.65 GHz. The XDM Digital Back-End[DBE] system was tasked with sampling and processing the signal. The DBE was planned to consist of several XDM Carrier Card onto which attached XDM Digital Receiver cards.

The XCC was required to comply to the planned XDM DBE physical architecture and thus had the following physical requirements:

- The board was to take a PXI Express form factor.

- The board was to include two Ten Gigabit Ethernet CX4 links.
- A programmable logic device was to be included for communications tasks.
- A controller module capable of running Linux was to be included.
- A system health monitoring device was to be included.

The scope of this project included designing and implementing the XCC's hardware and gateware to a point where the functionality required for XDM would be achieved.

1.2.2 Chapter 3: Hardware Design

The requirement for the XCC specified several physical hardware elements to be included on the board. This chapter starts by presenting the specific components and interfaces chosen to satisfy the requirements. The chosen hardware design was comprised of the following primary elements:

- Two Xilinx Virtex-4 FX60s, chosen for communications tasks
- An Avnet Mini-Module Virtex-4 FX12-based controller module
- An Actel Fusion FPGA, for health monitoring
- A serial, point-to-point control interface

The schematics were captured using a hierarchical system with the Mentor Graphics package, DxDesigner. The FPGA I/O allocation was performed using the Mentor Graphics tool, IODesigner. The PCB layout was contracted in an attempt to accelerate the development time of the board. A set of design rules was developed as an input to the layout development.

Signal integrity was a concern throughout the design. A software package called HyperLynx was used to simulate transmission lines during both the design and layout phases. The package was capable of running simulation of an imported PCB layout, thus providing a high degree of certainty that a physical line would work.

1.2.3 Chapter 4: Gateware Design

The XCC included four FPGAs, each of which required *gateware* to perform their intended functions. Three gateware designs were needed: One for the Actel Fusion FPGA, one for the FX12 controller FPGA and one for both Xilinx Virtex-4 FX60s.

The Actel Fusion gateware, which is referred to as the XCC Power Supervisor gateware, was designed to perform system health monitoring and power management functions. The primary function of the

design was to interface with the Fusion's Analogue Block, which included an ADC with numerous configurations. System health information was obtained through values sampled by the ADC. An I^2C interface provided the controller with access to this health information.

The Virtex-4 FX12 gateware, referred to as the XCC Controller gateware, was designed to map the resources of the XCC to memory on the PowerPC. This was achieved using the Xilinx Embedded Development Kit[EDK]. The following controllers were attached to the PowerPC's memory map:

- Xilinx DDR SDRAM, Flash memory and Bootable ROM controllers
- OpenCores I^2C and SPI controllers
- Custom SelectMap and Control Interface controllers

The serial, point-to-point control interface required a complicated gateware solution in order to operate. The gateware system that was developed included the following features:

- Automatic bit- and byte-alignment of the data, using Virtex-4 ISerDes resources
- Direct memory-mapped access to slave resources
- Indirect, high-efficiency bulk transfers from slave memory to a local buffer

The Virtex-4 FX60 gateware, referred to as the XCC Communications FPGA gateware, was tasked with handling communications with the XMCs and the Ten Gigabit Ethernet links. However, no XMC cards were available at the time of completion of this project, so this interface could not be tested. Two modules were used to communicate over the CX4 ports. The first module, called the XAUI Pipe, was a simple module for point-to-point communications and was developed primarily for testing. The second module was a modified version of a University of California, Berkeley Ten Gigabit Ethernet controller. This module included facilities to communicate, using MAC frames and UDP/IP packets.

1.2.4 Chapter 5: System Verification and Performance

This chapter describes the testing and verification procedures and reports on the board performance.

The first tests were carried out with the power off-line. These included short circuit and power supply connector pinout checks. The ATX power connector had the wrong pin ordering and a ATX power supply cable was modified accordingly.

The next set of tests were carried out on current-limited auxiliary power. The auxiliary power system worked as intended. The Fusion FPGA was programmed successfully and the serial communications were shown to operate without error.

The next tests were carried out with full power. A minor error with the power supply chain was solved and the board was successfully brought to the powered-up state. The only power-supply that did not work was the DDR2 memory termination. This was left disabled as the DDR2 memory was not tested.

The XCC Controller was successfully programmed and no problems were encountered with the design. The Control Interface exhibited a BER of less than 10^{-10}. The performance of the bus on the Control Interface was lower than expected and a DMA controller would be necessary in future should performance be required.

Eye diagrams for the Ten Gigabit Ethernet were measured using a Tektronix DSA 70804 digital serial analyzer and were sufficiently open at cable lengths of 5 metres. A counter was then transmitted from one CX4 link to the other. No errors were encountered in the transfer of ten terabits of data, which amounted to a BER of less than 10^{-13}. The final Ten Gigabit Ethernet test involved using a Ten Gigabit Ethernet core to transmit UDP packets to a Myricom network adaptor, plugged into a Dell server. The link was found to operate successfully up to 7.5 Gbps.

1.2.5 Chapter 6: Conclusion and Further Work

This chapter first presents a summary of the performance of the system and then discusses future work and possible improvements.

A primary function of the board, the communication between XCC Communication FPGA and the XMC cards, could not be tested, owing to the absence of a daughter card at the time of completion of this dissertation. However, all other functionality requirements for XDM were met:

- Error-free Ten Gigabit Ethernet communications
- Control communications with a bulk read throughput of close to 400 Mbps
- FPGA configuration
- Power management and supply

The following items are elements on the XCC that require further work:

- The DDR2 memory is untested and requires a gateware core to be developed for testing and integration.
- The backplane clock distribution and power supply were not implemented and require testing.
- The XMC interface on the XCC Communication FPGA needs to be tested.

- The XCC Controller requires a DMA core to improve system performance.
- The SDCard interface has not been tested.
- The SelectMap configuration interface has not been tested.
- The gateware controlling the ADC on the Actel Fusion needs to be corrected to support current and temperature monitoring.

2 Requirements and Specifications

The XDM Carrier Card was designed to be part of the XDM Digital Back-End [DBE]. Thus, most of the XCC's requirements were derived from those of the XDM DBE. This chapter describes the planned XDM DBE and presents the design requirements for the XCC. [1]

2.1 XDM Description

The XDM was the first prototype of the Karoo Array Telescope and was comprised of a single dish radio telescope with a bandwidth from 1.414 to 1.65 GHz. The primary goal of the XDM was to serve as a platform to advance the software, hardware and system architecture for the much larger MeerKAT.

The XDM consists of a single dish with a dual-polarized, seven-horn cluster feed. The Radio-Frequency subsystem performs amplification, among its other operations, and transfers the signal from the dish to the processing room, using RF-over-fibre technology. The signals are then digitized and processed by the Digital-Back End. Thereafter, the processed signal is transmitted to the computing infrastructure for further off-line processing.

2.2 XDM DBE Overview

A summary of the planned DBE architecture for XDM is shown in Figure 2. It was planned that the system would include a sub-rack with power supplies, cooling, a master controller computer and two XDM Carrier Cards, each carrying two XDM Digital Receiver Cards [XDRs]. The analogue signals were to be directly sampled and processed on the XDRs. The processed data would then be passed to the carrier card, which would then transfer it on to a high-speed 10 Gigabit Ethernet network. Each XCC would include a controller module, running Linux, to perform control, configuration and monitoring as necessary. [Owing to time constraints, the DBE architecture was abandoned for the purpose of XDM. However, work relating to this project continued as an academic exercise.]

2.3 Hardware Requirements

2.3.1 Compact PXI Express Compatibility

The XCC was required to take a 6U, 23.34mm x 16.0mm, EuroCard form factor and to include features to support operating as a PXI peripheral card. These features included both the necessary

[1] All details regarding XDM DBE requirements and specification were established through internal communication

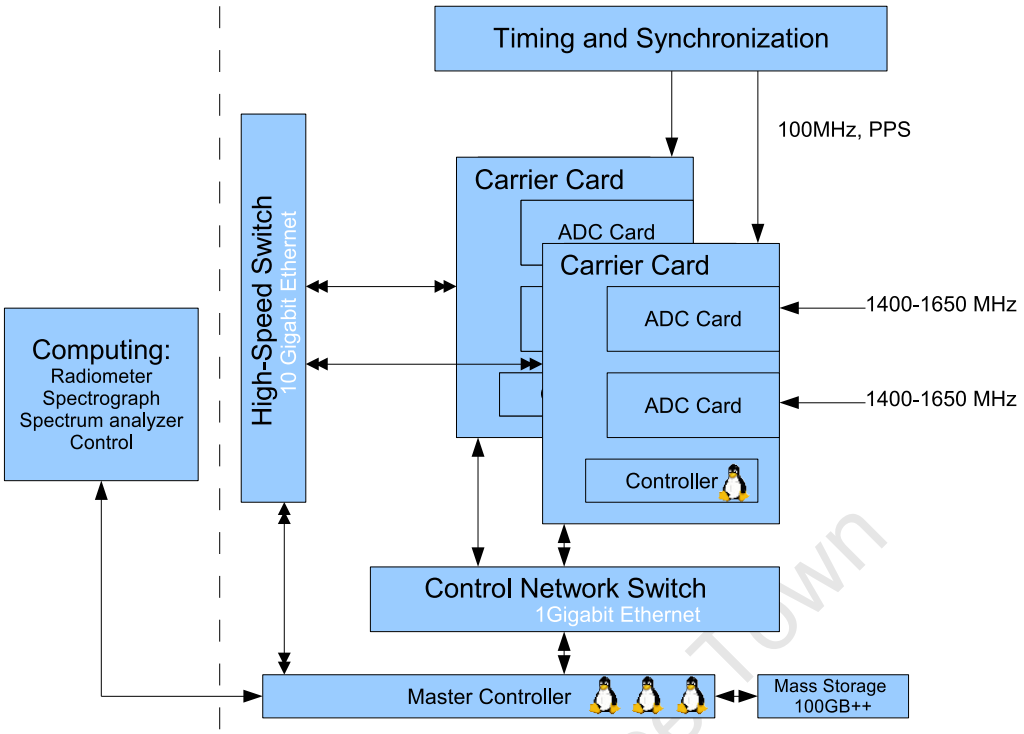


Figure 2: XDM Digital Back-End architecture

connectors and electronics to allow full PXI operation. The PXI Express backplane provided power and synchronization functions.

2.3.2 XMC Mezzanine Card Interfaces

The XCC was required to support two XMC mezzanine cards. XMC is a standard mezzanine card interface, which includes: [1]

- 40 high-speed differential pairs
- 40 user-definable, single-ended I/Os
- JTAG
- I^2C
- Power

In XDM, these cards would take the form of digital receiver cards. However, the functionality of mezzanine cards could vary if they included a compatible XMC interface. The XMC specification placed restrictions on component height between mother and daughter cards. Component placement had to satisfy these restrictions.

2.3.3 Ten Gigabit Ethernet

The XCC was required to include at least two 10GBASE-CX4 links. These links were required to transfer data between the XMC cards and a high-speed data network.

2.3.4 Control Network

A single 10/100/1000 Ethernet connection was required as a communications channel between a remote operator and any control software running on the XCC.

2.3.5 DDR2 SDRAM Memory

A large store of DDR2 SDRAM memory was set as a requirement for the XCC. DDR2 memory was the only current technology that provided sufficient speed and capacity to store incoming data. This memory was included to perform buffering for transient analysis and to support DSP if necessary. The memory was not required to achieve XDM functionality.

2.3.6 Communications FPGA

An FPGA was required to perform the tasks necessary to transfer data between the high-speed network and the XMC cards. If necessary, the FPGA should be able to perform DSP functions. [This FPGA is referred to as the XCC Communications FPGA in this document.]

2.3.7 Controller Module

The XCC was required to include a controller that was capable of running an embedded version of the Linux operating system[OS]. The following functions needed to be implemented by the controller:

- Control of the elements on the board
- Configuration of the board's FPGAs
- Monitoring the health of the board
- Remote operation over the 10/100/1000 Ethernet link

The controller was required to be a computer-on-module. This would mitigate the risks associated with the design of controller circuitry.

2.3.8 Power Management and System Health Monitoring

KAT will be situated in the Karoo, a remote, semi-desert region. It is imperative that any system deployed there be extremely reliable. For this reason, a way of measuring and monitoring the health of the XCC was required. This system was required to change the power state of the board, given thresholds for voltages, temperatures and currents, with the goal of turning off the board, if necessary, to avoid damage to the on-board electronics. Further, problems with the board should be easily diagnosed so that low-skilled technicians can be instructed which board to remove or replace.

2.4 Scope of Work

The work for this project included the design and development of the hardware of the XCC and the development of gateware to perform the basic tasks required for XDM. In addition, software was to be developed to test the functionality of the hardware and gateware.

Several hardware elements were untested, owing to either the lack of availability of components or to their not being required for XDM. Some software-oriented tasks, such as the development of a Linux port, were not within the scope of work of this project.

2.5 Deliverables

The primary hardware deliverables for this project included:

- Hardware schematics
- Printed Circuit Board [PCB] layout files
- Tested hardware

The gateware deliverables included HDL code, organized in projects for each programmable logic device. Automated build and testing environments were included for each project.

2.6 Conclusion

In this chapter, the requirements for the XCC, both functional and physical, were discussed. The physical requirements were set as follows:

- The XCC needed to be a 6U Compact PXI Express compatible card.

- It was required that two XMC mezzanine card interfaces were included to support two daughter cards.
- Two Ten Gigabit Ethernet interfaces were required for data communications.
- A single 10/100 Ethernet link was required for control communications.
- A DDR2 SDRAM bank with high-speed and high-volume capabilities was required.
- A reconfigurable logic device, called the XCC Communication Logic, was required to perform communications tasks.
- A controller, called the XCC Controller, capable of running an embedded operating system, was required.
- A programmable device, called the XCC Power Supervisor, was required to control power states and monitor system health.

The functions required to be implemented by the gateware were as follows:

- Power sequencing, health monitoring and serial communications were to be performed by the XCC Power Supervisor.
- The XCC Controller was to support control, monitoring and FPGA configuration capabilities.
- The XCC Communications FPGA was to relay data from the XMC cards to the Ten Gigabit Ethernet links.

The deliverables of the project were as follows:

- Hardware schematics, Printed Circuit Board [PCB] layout and final hardware
- Final gateware images, Hardware Description Language [HDL] files, test-benches and automated build infrastructure

3 Hardware Design and Implementation

The purpose of this chapter is to present the design and implementation of the XCC's hardware. First, there is a presentation of the the basic design architecture, as governed by the requirements. Next, there is a discussion regarding the major decisions that were reached in the final design of this architecture. Finally, the implementation of the XCC hardware is presented.

3.1 XCC Hardware Design

The XCC's basic architecture, to be obtained from the user requirements, is shown in Figure 3. The XCC consists of the following primary elements:

- A board controller module
- Two XMC mezzanine card connections
- DDR2 Memory
- XCC Communications FPGA
- Power monitoring and supervision

The internal and external electrical interfaces mentioned in the requirements are also summarized in Figure 3.

The external interfaces include:

- PXI Express power, power management, timing and synchronization signals
- Two CX4 Ten Gigabit Ethernet data network ports
- A 10/100/1000 Ethernet control network port

The internal interfaces include:

- A control interface that allows commands and data to be transferred between the controller and the XCC Communications FPGA, as well as slave devices plugged into the XMC slots
- A FPGA configuration bus that allows the controller to program the communications FPGA, or FPGAs on the mezzanine cards
- A System health monitoring bus that connects the controller to monitoring devices on the XCC and XMC cards
- A high-speed interface between both XMC slots and the communications FPGA

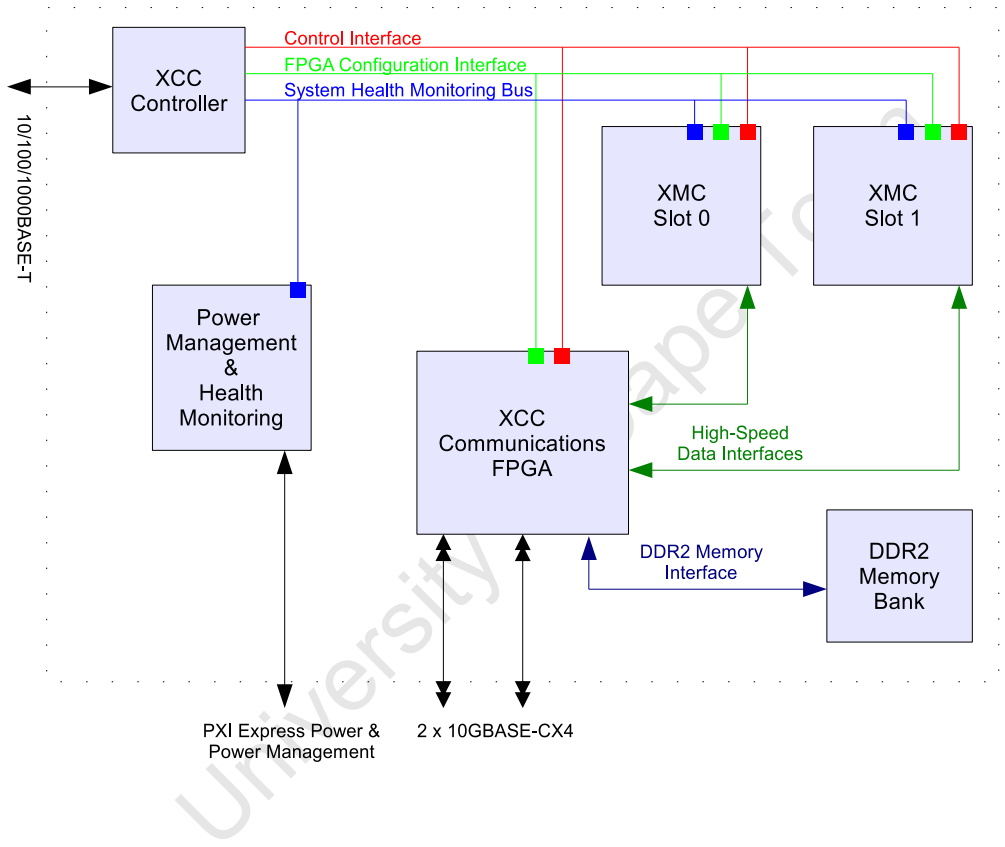


Figure 3: XCC Basic Architecture

Table 1: I/O Requirements of XCC Communications FPGA

Interface name	Number of I/Os used
XMC 0	88
XMC 1	88
DDR2 SODIMM	122
Control Interface	24
Clocks	8
GPIO	5
Voltage reference and termination	26
Total	361

3.1.1 XCC Communications FPGA

The communications FPGA on the XCC was required to include multi-gigabit SerDes [Serializer/Deserializers] to support Ten Gigabit Ethernet directly on the device. At the time of design only three FPGA families from major manufacturers satisfied this requirement. These were:

- Xilinx Virtex-4 FX FPGAs
- Altera Stratix-II GX FPGAs
- Lattice SC FPGAs

The Lattice FPGA provided the lowest cost option and included dedicated silicon for DDR memory and Ten Gigabit Ethernet controllers. However, this device was very early in its development cycle and posed a risk on the basis of availability and possible undocumented device limitations. Furthermore, the Lattice FPGA would not have enough resources to perform any DSP tasks should they be required. [2]

The Altera Stratix-II GX FPGAs support LVDS on limited I/O banks, which would have imposed limitations on I/O allocation and increased routing complexity. [3]

Virtex-4 FPGAs provided the lowest risk option for the design and were thus chosen.

The only widely available Virtex-4 FX series FPGA at the time of design was the Virtex-4 FX60 FF672 FPGA. This FPGA included 12 multi-gigabit transceivers and 352 user-definable I/Os. A summary of the I/O requirements for the communications logic is shown in Table 1. Two FX60 FF672 FPGAs were required to satisfy the I/O requirements.

The fastest way of programming Virtex-4 FPGAs was via an 8-bit wide, parallel configuration bus, called SelectMap. Therefore, the FPGA configuration bus was designed for SelectMap compatibility. JTAG was essential for testing and configuation and was included for all FPGAs on the XCC and XMCs. [4]

3.1.2 XCC Controller

The XCC controller module needed to include a processor, RAM, Flash Memory, an Ethernet Physical Layer chip [PHY], an Ethernet Media Access Control chip [MAC], I^2C for the System Health Monitoring bus and a local bus or enough GPIO to operate the FPGA configuration and control interface buses. The following computer modules satisfied the selection criteria and were considered:

- Avnet FX12 Mini-Module: An FPGA based controller with embedded PowerPC Core
- Compulab CM-X270: Intel XScale based controller
- SSV DIL/NetPC DNP/920: ARM9 Based processor

The controller was required to operate a control interface to each FX60 and each XMC. The performance of this interface had to saturate a 100 Mbps Ethernet link. In order to achieve this a bandwidth of 320 Mbps was used as the bus requirement. If a traditional 8-bit bus had been used this would have equated to a rate of 40 Mbps per data line. Figure 4 shows a simulated unterminated and unbuffered clock signal, running at 40 MHz, bussed to all control interface targets. This line configuration might have failed owing to signal integrity problems caused by the length of traces and the stubs introduced by the XMCs. The possible solutions to this problem were as follows:

- Increase the bus width and slow the data rate per line
- Reduce the stub-length by buffering the bus
- Implement a serial, point-to-point protocol

The first solution was not desirable as it would have increased routing complexity and also would have increased I/O utilization on FPGAs. The second solution was not chosen as it would have introduced additional components and routing. A serial point-to-point protocol implemented with Low-Voltage Differential Swing[LVDS] logic had the following benefits: [9]

- Correctly terminated LVDS lines could be routed to lengths of tens of centimetres without exhibiting signal integrity problems.
- LVDS exhibited high common-mode rejection, making it more immune to interference such as cross talk.
- The low-voltage swing greatly reduced the radiated power.
- A signal LVDS pair could operate up to 1 Gbps on the highest speed grade Virtex-4 devices. [4]

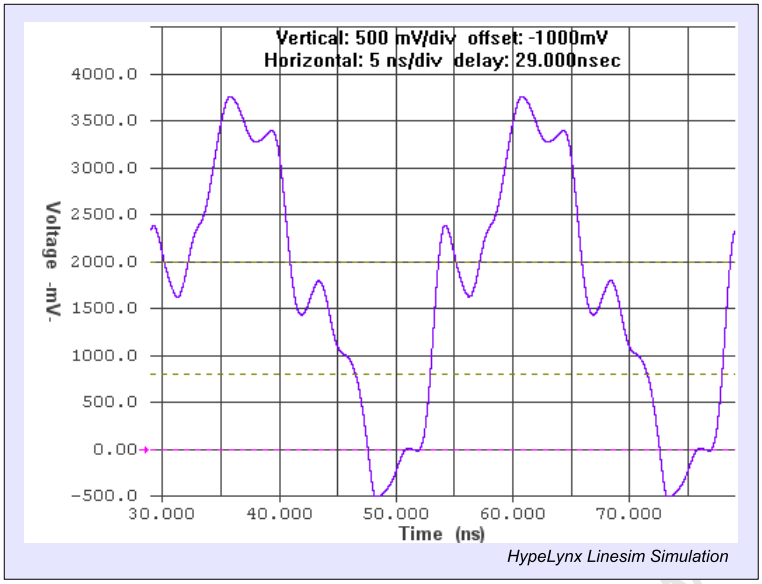


Figure 4: Simulated electrical performance of bussed control interface

The only controller that was capable of implementing such a protocol was the Avnet Mini-Module, which was duly chosen. For full specifications of this module refer to the Avnet Mini-Module user guide. [5]

In order to run a full operating system, it was necessary to include enough Flash memory for both the kernel and a filesystem. The 4 MB Flash memory on the Avnet Mini-Module was large enough for a kernel and a small filesystem. However, a miniSD Card slot was included in the design to provide enough non-volatile memory to support a large filesystem.

3.1.3 Power Supervisor

The power supervisor had to be capable of controlling the power state of the board, taking into account both controller and system health inputs. Furthermore, the device needed to provide the XCC Controller with access to system health information, such as voltages, temperatures and currents. Two devices were considered to perform this task:

- Analog Devices ADM1063 Super Sequencer
- Actel Fusion AFS600 FPGA

The ADM1063 is an IC that includes a 12-bit ADC and a programmable power-sequencing engine. This device is capable of measuring up to 10 voltages and two temperatures and includes several configurable outputs and an I^2C interface. The Actel Fusion is a system-on-chip, mixed-signal FPGA,

with embedded ADC, Flash memory, clocks and voltage regulator. [6] This device, while not as low-cost as the ADM1063, includes far more capabilities, which allow it to perform various additional tasks, such as:

- Monitoring of up to 30 analogue sources
- Support for multiple communications interfaces, such as backplane I^2C, controller I^2C and serial port
- General purpose tasks, such as conditional JTAG forwarding and configuring clock buffs

Conditional JTAG forwarding was required in order to remove non-present XMC cards from the chain. The AD9511 clock buffers required configuration via an SPI interface, in order to correctly buffer the back-plane clocks. The Fusion FPGA was chosen owing to its ability to perform these additional tasks.

3.1.4 Clock Generation and Distribution

The primary clock source was a 100MHz LVDS signal that was sourced from the backplane. This clock signal was buffered, using an Analog Devices AD9511, which claimed to introduce less than one picosecond jitter. This low jitter was required to allow the clock to be used as a sampling clock for ADCs on the XMCs. The AD9511 outputs five copies of the input clock to each FX60, each XMC and to the controller module. The same distribution system was used for the one Pulse-Per-Second[PPS] synchronization signal from the backplane.

Each FPGA was set up to include its own clock source so that it could be operated independently of the backplane during testing. The controller module included a 100 MHz clock. Each FX60 attached to a 200 MHz clock source. The Actel Fusion FPGA included an embedded 100 MHz oscillator and also was attached to a 10 MHz crystal to operate a Real-Time Clock [RTC].

The Multi-Gigabit Tranceivers [MGTs] on the Virtex-4s required a low jitter 156.25 MHz reference clock to support Ten Gigabit Ethernet. This clock was generated from a single Surface Acoustic Wave[SAW] clock oscillator and was buffered with a Micrel SY89832U 1:4 clock buffer and distributed to each FX60. [8]

3.1.5 Power Generation

The primary power source was designed to be the PXI Express backplane, which supplied 12V, 5V, 3.3V and a 5V auxiliary power. An addition, an ATX power supply was added to the board to facilitate bench operation and it included the same supplies. The always-on 5V auxiliary supply was used to

Table 2: Estimated Power Requirements for the XCC and two XDRs

Supply name	Power Required [Watts]
0.9V Termination	3.8
1.2V	14.32
1.2V MGT	6.2
1.5V	1.44
1.8V	5.78
2.5V	6.47
5V Aux	4.74
3.3V Backplane	10.75
5V Backplane	19.5
12V Backplane	21.12
Total	94.12

power the Actel Fusion FPGA. The Actel Fusion FPGA, as part of its power-sequencing operation, would enable the 12V, 5V and 3.3V supplies.

The estimated power requirements of the XCC and two XDRs are shown in Table 2. These values were derived from several sources, including Xilinx's Virtex-4 power estimator and a Micrel DDR2 design guide. The supplies with the highest power ratings, 1.2V, 1.8V and 2.5V, required efficient DC-DC converters. Texas Instruments produces a line of switch-mode power supply module in their POLA range that operate at up to 90% efficiency. These devices provided a compact solution for designs requiring high-power and were thus chosen for the XCC. The other supplies were generated using various Texas Instruments linear regulators.

3.1.6 Thermal Considerations

The Xilinx Virtex-4 XPower Estimator tool was used to estimate the thermal cooling requirement of a fully utilized FX60 FPGA. The tool estimated that, with an ambient temperature of 20°C, the heatsink and airflow would need to provide a thermal resistance of 3.3°C/W. Standard isolated heatsinks would not provide a low enough thermal resistance, as airflow would be disrupted by the XMC cards. The only feasible way of achieving the necessary cooling would be to use a large surface-area integrated heatsink. Thus, mounting holes were included to support an integrated heatsink.

3.1.7 Detailed Design Overview

The complete design as discussed is shown in Figure 5. This diagram indicates the component and interface details.

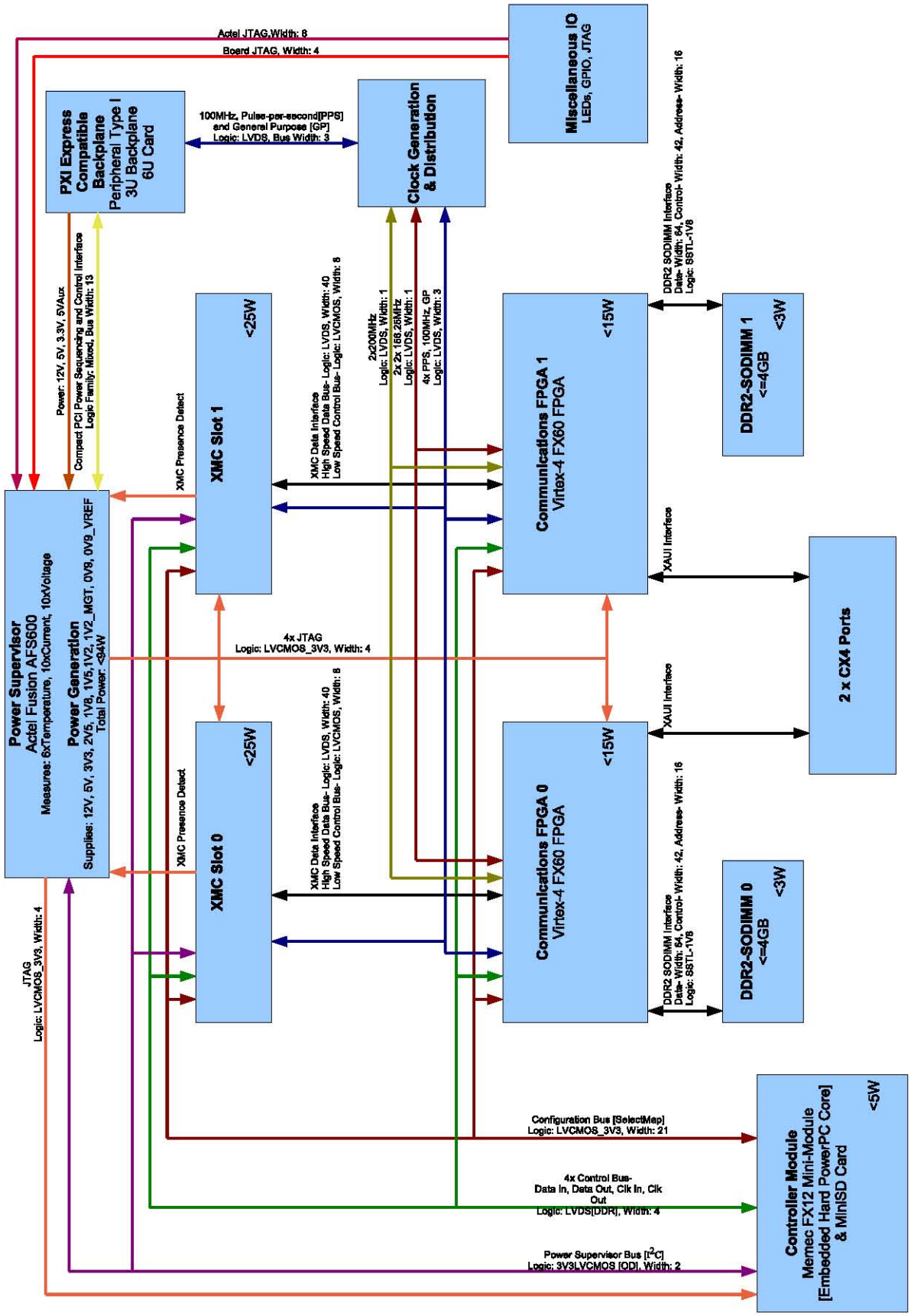


Figure 5: Detailed XCC Block Diagram

3.2 Schematic Design Capture

Hardware schematics are documents that contain information about the components included in a design and their interconnections. Schematics are used as an input to PCB layout and are useful as a reference when utilizing the hardware. A hierarchical approach to schematic capture was used in this project. Hierarchical schematics include layers of interconnected blocks, where the lowest layer blocks are populated with components. This approach has several design benefits over a flat schematic system, including improvements in manageability, problem subdivision and design re-use. The gains are similar to those of modular software coding.

3.2.1 FPGA I/O Allocation

FPGAs include I/O banks that must be configured to suit the design in which they are being used. Designing the allocation of I/Os can be a difficult process, as there are design rules that need to be obeyed for various logic families. Failure to observe the correct rules can cause errors in or failure of the design. It is also inevitable during the design of FPGA-based hardware that several iterations of I/O reallocation will be required to minimize the crossing of lines encountered with PCB layout.

Mentor Graphics has a product, called IODesigner, that aims to streamline the process of assigning FPGA I/Os. This package provides I/O bank design-rule checking, schematic symbol generation and links to the HDL development. This package was used to accelerate the allocation of I/O Banks on the Virtex-4 FX60s and Actel Fusion FPGAs.

3.2.2 XCC Schematics

The XCC schematics are included on the attached CD. Refer to Appendix A for the location of the schematics of the XCC.

3.3 Printed Circuit Board Layout

The printed circuit board layout of the XCC was subcontracted to a Cape Town-based company called Peralex. This step was taken to accelerate the development speed by utilizing their PCB layout experience and allowed the gateware to be developed concurrently.

3.3.1 Layout Constraints

Almost all signal routing was performed, using an auto-router. Auto-routers route the traces between components while trying to adhere to a set of layout constraints. Such a set of constraints were pro-

vided as an input to Peralex for the layout. The items specified as layout constraints are summarized as follows:

- Priority: Higher priority traces are routed with more effort to achieve their constraints.
- Differential: Traces are routed differentially.
- Group: Traces are routed in groups to optimize skew within related buses.
- Maximum and minimum trace lengths: The length of a trace affects signal integrity and skew within a group.
- Impedance: Electrically long traces require impedance control to minimize reflections.
- Stub control: Bussed lines require the stubs off the bus to be minimized to reduce reflections.

3.3.2 Completed PCB Layout

All files relating to the PCB layout, manufacturing and assembly are provided on the attached CD and are summarized in Appendix A.

3.4 Signal Integrity Simulations

Signal integrity and Electro-Magnetic Interference[EMI] were deemed high risk items and were kept in mind throughout the hardware design and implementation. A software package from Mentor Graphics, called HyperLynx, was used to simulate the signal integrity and EMI performance of the XCC. The software included two packages. The first, called LineSim, simulated a schematic entry transmission line. The second, called BoardSim, simulated the SI performance of a completed PCB layout.

LineSim was used during the design phase to simulate the transmission lines that were expected to perform poorly from an signal integrity perspective. BoardSim was used to simulate the performance of the final layout. It was found that the autorouter successfully reduced crosstalk and matched the target impedances.

The LineSim and BoardSim simulation results are attached in Appendix B. The validity of the results relied on the correctness of the models and assumptions made about connector performance. Critically, the performance of the XAUI could not be accurately simulated, owing to the lack of a Virtex-4 MGT simulation model.

3.5 Completed Hardware

Figure 6 show images of the completed PCB without components. Figure 7 shows the XCC populated with all components excluding the PXI backplane connectors.

3.6 Conclusion

The final hardware design included:

- Two Xilinx Virtex-4 FX60 FPGAs for XCC Communications Logic
- An Avnet controller module with a Xilinx FX12 FPGA with embedded PowerPC, selected as the XCC Controller
- An Actel Fusion FPGA with embedded ADC for power supervision and monitoring, selected as the XCC Power Supervisor
- A high-speed point-to-point LVDS control interface between the XCC Controller and slave FPGAs
- A high-speed point-to-point LVDS interface between XMCs and XCC Communications FPGAs
- A parallel FPGA configuration bus from the XCC Controller to each slave FPGA

The schematics were captured using a hierarchical scheme. The FPGA I/O allocation was performed with the aid of a tool called IODesigner from Mentor Graphics. The Printed Circuit Board layout was outsourced as a means of accelerating the development of the system. A set of layout rules was developed and used as an input to the layout contractors. These constraints were to ensure correct electrical behaviour of the signals specified therein and to ensure that signal integrity did not degrade the ultimate performance of the board.

Signal integrity was a major concern throughout the hardware design process. A software package, HyperLynx, was used to simulate signal integrity during both the hardware design and implementation phases. The results of these simulations were the basis of several design decisions and were used to verify the layout.

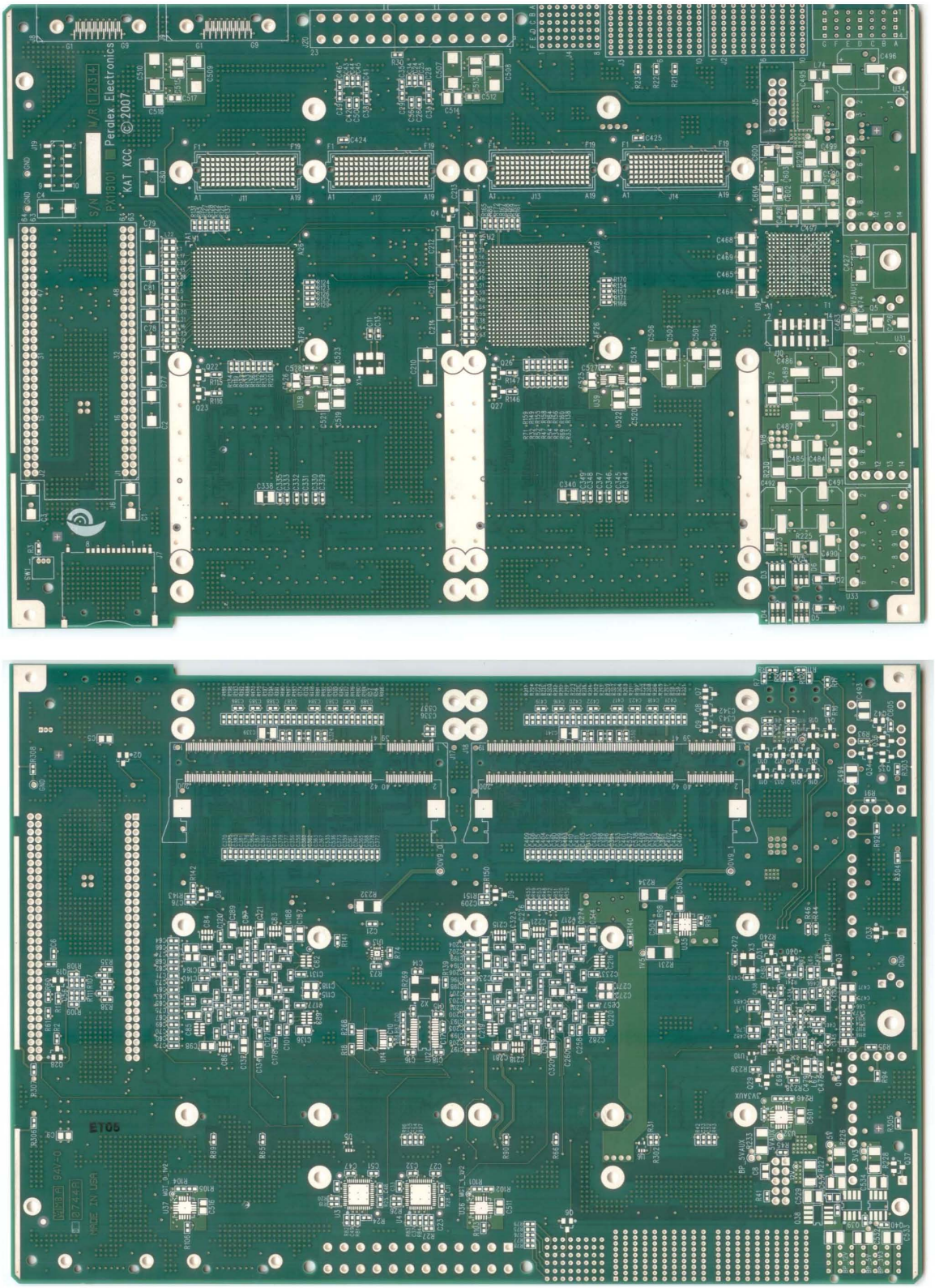


Figure 6: XCC printed circuit board: top and bottom sides

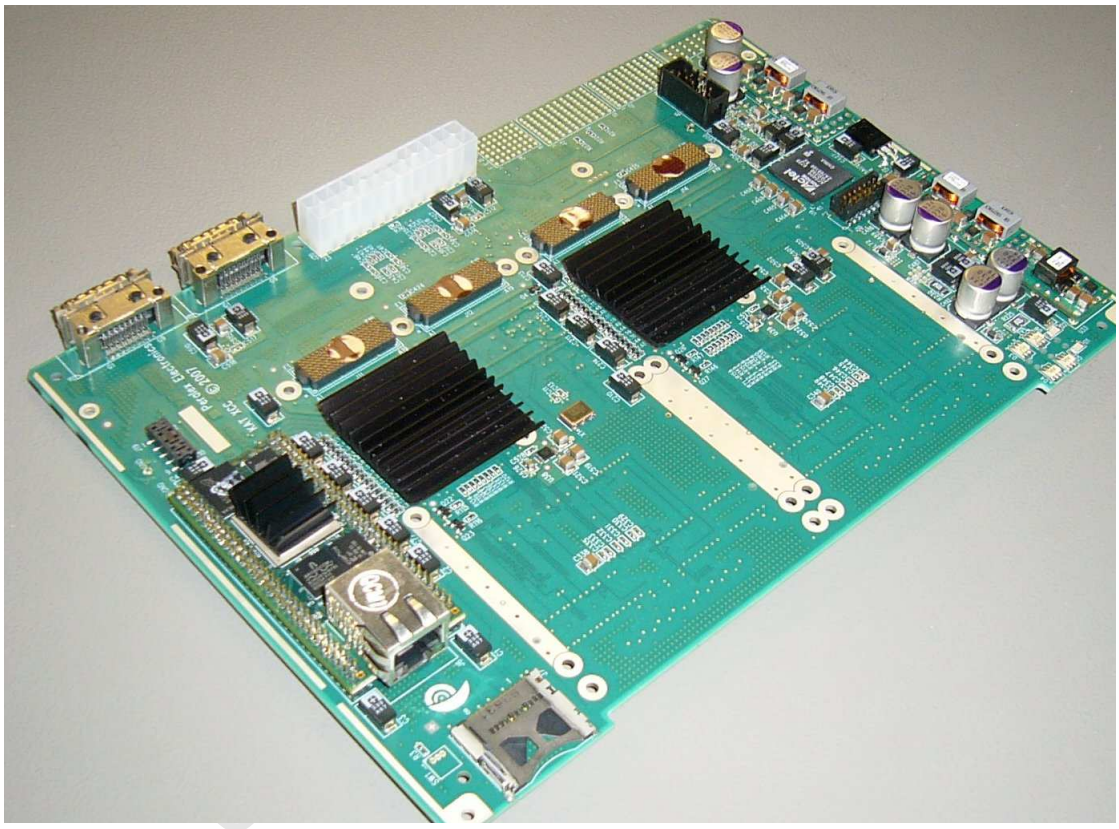


Figure 7: XCC populated with all components excluding the PXI backplane connectors

4 Gateware Design

The XCC includes four FPGAs, each of which require gateware to perform their intended functions. Three gateware designs were required: One for the Actel Fusion FPGA, one for the FX12 controller FPGA and one for both Xilinx Virtex-4 FX60s. The Actel Fusion gateware, which is referred to as the XCC Power Supervisor gateware, performs system health monitoring and power management functions. The XCC Controller gateware provides interfaces to software that is running on the embedded PowerPC, while the XCC Communications FPGA provides Ten Gigabit Ethernet communications to the XMC cards.

4.1 XCC Power Supervisor

4.1.1 Device Capabilities

The Actel Fusion AFS600 FPGA included an analogue circuitry module, embedded on the FPGA, called the Analogue Block. The primary function of the Analogue Block was to sample analogue inputs with its built-in ADC. The ADC could sample 12-bits at 400 kHz and included a multiplexer that could select up to 30 external analogue inputs and two internal inputs, which included the die temperature and the FPGA core voltage. The external signals could be connected directly to the ADC or through prescaler elements, which scaled the input voltages to meet the range of the ADC. It was also possible to configure the device to support measuring of up to ten currents and temperatures. The Analogue Block included FET gate drivers that could be used to control power MOSFET switches.

The Analogue Block supported numerous configurations for analogue sources, gate drivers and the ADC. Many of these were configurable at run time through the Analogue Configuration Mux [ACM], including:

- Type of source: voltage, current or temperature
- Pre-scale before ADC input
- FET gate driver control

The Fusion FPGA included two types of Flash memories, which were accessible through the FPGA fabric. The first was a 4 Mb memory called the Flash Memory Block. This memory allowed read and write access through the fabric. The second memory block was called the FlashROM. This was a 128-byte, read-only memory that was configurable only via JTAG.

The FPGA included an embedded 200 MHz RC oscillator and thus required no external clocks. In addition, the Actel Fusion supported an external crystal that could be used to drive an internal Real-Time Clock [RTC]. This RTC had the capacity to power down the FPGA for a given number of

cycles. The Actel also included several features common to most FPGA families, such as internal SRAM memory blocks and configurable I/Os.

4.1.2 Design Overview

Figure 8 shows the interfaces to the Actel Fusion FPGA on the XCC. The functions that the Actel Fusion gateware performed are summarized as follows:

- The management of power sequencing
- The measurement of all attached analogue inputs
- The performance of automated responses to various analogue input levels, to prevent potential component damage
- The provision to the controller of access to its internal registers via I^2C
- The management of the board JTAG chain
- The configuration of the AD9511 clock buffers

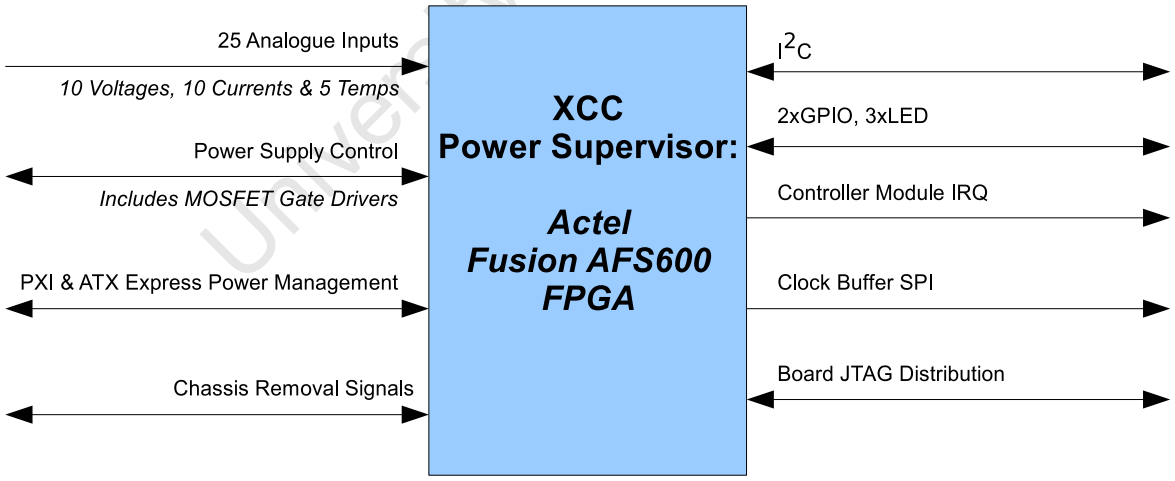


Figure 8: Actel Fusion Interface Diagram

The design is described in detail in Figure 9. The design included 16 Verilog HDL-defined modules. Several of these modules in turn included memory-mapped registers that were accessible via a bus called the LBus.

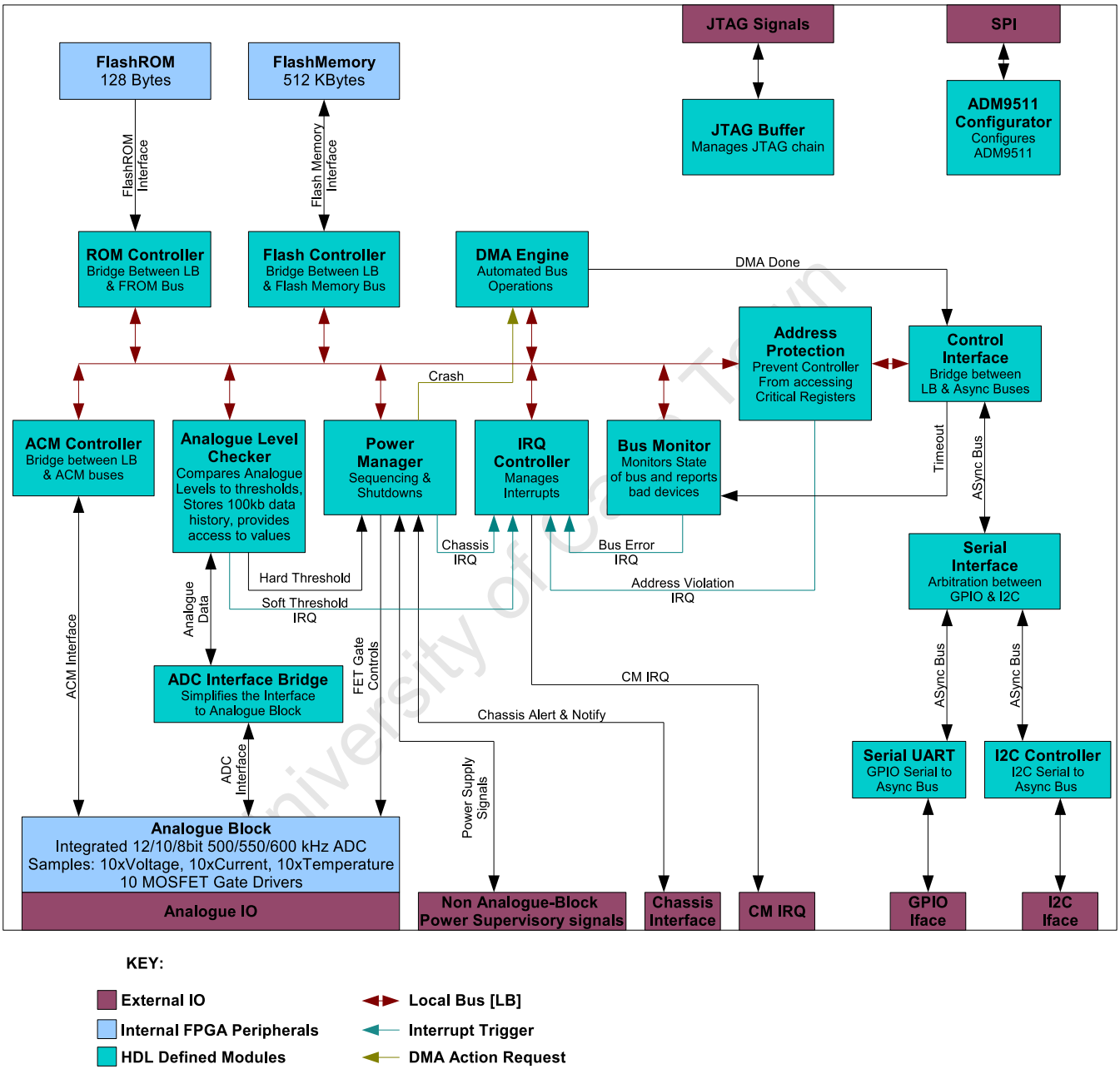


Figure 9: Actel Fusion Gateware Block Diagram

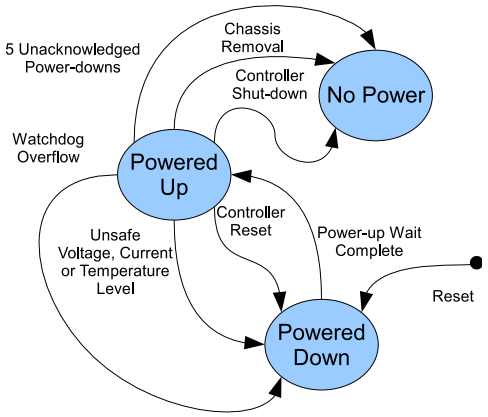


Figure 10: XCC Power Supervisor Power State Machine

4.1.3 Power Sequencing

The Power Manager module controlled the power state of the XCC. The state machine is described in Figure 10. The XCC Power Supervisor came out of reset in the *powered-down* state. In this state, all power supplies on the XCC were disabled. After a delay that was based on the Compact PXI geographical address, the Power Manager entered the *powered-up* state. In this state, all power supplies were enabled and regular board functions could be performed.

The state would change to *powered-down* under the following conditions:

1. If the controller module issued a reset command
2. If an internal, configurable watchdog timer was not reset
3. If an unsafe voltage, current or temperature condition was detected

One second later, the state would return to *powered-up*. If conditions *2* or *3* occurred five times without the controller acknowledging the crashes, the board would enter the *no-power* state. In this state, all power supplies would be disabled and the only means of entering the *powered-up* state again would be to reset the FPGA via the serial port or to power-cycle the board. The no-power state could also be entered if the controller issued a power-down command or after the chassis removal alert had been triggered and acknowledged.

4.1.4 Analogue Value Measurement

The ADC was configured at build-time to sample 12-bits at 100 kHz. The Fusion's Analogue Block needed to be configured at run-time before analogue signals could be sampled. This was done by

setting registers on the ACM Controller module. After the Analogue Block had been configured, all 32 possible analogue sources were sampled in sequence and stored in registers in the Analogue Level Checker module. All samples were also stored in a 108 kb ring buffer, which was, likewise, accessible through registers. This provided storage of 9216 samples, which was equivalent to 92.16 ms of data. During a crash, this buffer would be stored into Flash memory and would provide detail as to the cause of the crash event.

4.1.5 Unsafe Analogue Level Responses

The checking of whether a sampled value was safe or not was performed by the Analogue Level Checker module. Two types of thresholds were defined: hard and soft. If a sampled value exceeded a soft threshold, an interrupt was triggered that notified the controller module. If a hard threshold was exceeded, the Power Manager was notified and the board was reset. The threshold level for each analogue input, both high and low, was configurable through 64 registers on the LBus.

4.1.6 Communications with the XCC Controller

The LBus was made accessible to the XCC controller through an I^2C interface. The byte-oriented I^2C interface was bridged on to the memory-mapped LBus by the Control Interface module. The memory operations were filtered by the Address Protection module. This prevented the controller from altering registers that might cause damage to the board, such as the hard threshold level registers.

The XCC Power Supervisor included an interrupt output that connected to the controller module. There were four conditions that could cause an interrupt to be triggered. These were:

- A soft threshold violation from the Analogue Level Checker
- A chassis removal signal
- A bus timeout from the Bus Monitor module
- An address violation from the Address Protection module

The IRQ Controller module managed the control and generation of interrupts.

4.1.7 Automatic Bus Operations

It was necessary that various LBus transactions be completed automatically in order for the XCC Power Supervisor to configure itself before entering the powered-up state. These automatic transactions were performed by the DMA Engine module. During these transactions, the LBus was locked and external transactions could not occur. The DMA Engine performed two operations:

- Startup configuration
- Crash detail logging

During the first step of startup configuration, the threshold values were read from the Flash ROM and were written to the Analogue Level Checker. During the second step, the Analogue Block configuration was read from the Flash ROM and written to the ACM. The crash detail logging involved reading data out of the Analogue Level Checker ring buffer and writing them to the Flash memory. This allowed for off-line analysis of the cause of a crash.

4.1.8 Miscellaneous Board Operations

The XCC Power Supervisor was capable of conditionally setting up JTAG chain of the board, thus allowing absent XMC cards to be removed from the chain. The gateware also included a write-only SPI interface for configuring the Analog Devices ADM9511 clock buffers.

4.1.9 Pre-Integration Testing and Results

The Actel Fusion gateware was tested on a development board called the Actel Fusion Starter Kit Board. This board included a Fusion AFS600 FPGA. An RS232 link was added to the board to provide communications to a PC. A small C program, called *Monman*, was developed to access the LBus registers via the link on a PC through a bridge, in a manner similar to how the controller had been designed to access registers.

Monman read commands from a file, transmitted them over the serial link and outputted the results of reads. A command typically consisted of a write or read to a memory address on the remote bus. However, there was support for other special commands, such as delays. This scheme allowed the design to be tested independently of the final hardware and reduced the integration time.

4.2 XCC Controller

The XCC Controller gateware and firmware was begun by two KAT team members, Andrew Martens and Marc Welz. This section describes the work done to extend their work and to test the gateware for the XCC Controller.

4.2.1 Controller Design on Xilinx FPGAs

The primary function of the gateware on the XCC Controller was to support the software running on the embedded PowerPC. Xilinx provided a tool, Embedded Development Kit [EDK], which was

designed to streamline the process of running software on either MicroBlaze- or PowerPC FPGA-based controllers. EDK linked the development of gateware attached to the processor with that of the source code to be run on the processor. The tool allowed the user to customize the memory map of the processor, using either user-defined gateware modules or cores from EDK libraries.

EDK used the GCC library suite to cross-compile and link user source code that was based on the gateware memory map. The tool generated an FPGA bitfile that initialized embedded FPGA memory with the compiled application. This allowed the user application to boot automatically.

4.2.2 Established Progress with Controller Module

The following had been completed before work relating to this project began:

- A working EDK project that included a DDR memory controller, a Flash Memory controller, Ethernet MAC and Serial UARTs
- A bootloader, Carabas, which provided Flash memory, network, debugging and application-booting mechanisms
- A working Linux port, including serial port and network drivers

4.2.3 Design Overview

The XCC Controller interfaces are shown in Figure 11. As per the scope of this project, the XCC Controller gateware was required to perform the following functions:

- Provide the embedded software with a means of communicating with the XCC Power Supervisor and any other devices on the Power Supervisor I^2C Bus.
- Provide the embedded software with a means of programming the FPGAs on the SelectMap bus
- Provide the embedded software with a means of communicating with devices using the Control Interface
- Provide the embedded software with a means of accessing the SDCard Device

Figure 12 presents an overview of the gateware developed for the XCC Controller. PLB and OPB are bus standards used by IBM processors and was used to attach many of the gateware modules to the PowerPC.

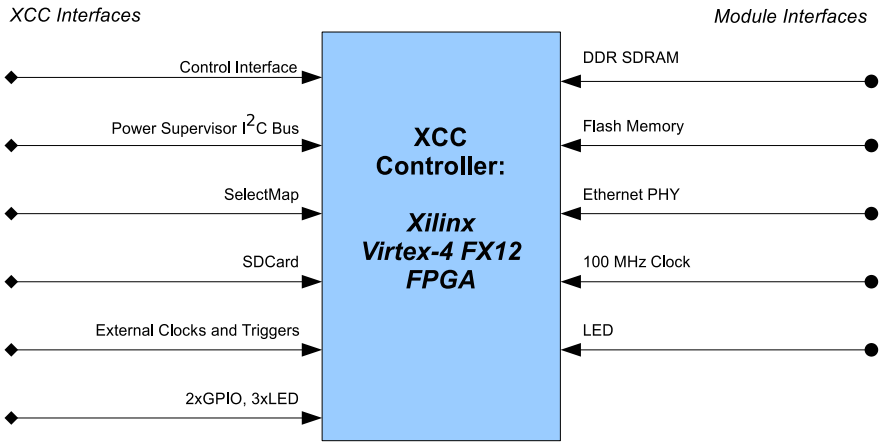


Figure 11: XCC Controller Interfaces

4.2.4 OpenCores WishBone Bus

OpenCores is a project that aims to provide open source, HDL-defined cores to the public by promoting the same community development principles that have made the Free Software movement successful. There are currently 435 projects hosted online by OpenCores that cover a wide range of applications, including Digital Signal Processing, communications and microprocessors. It is advantageous to use established open source cores for several reasons, which include:

- The reduction of time and financial costs of developing cores from scratch
- The elimination of the dependence on IP cores, which are often expensive and restrictively licensed
- The increased reliability, owing to the core being used within the community

The WishBone bus is an open source computer bus that is intended to connect cores within an IC, such as those on an FPGA. This bus is used by many OpenCores projects and was chosen as the desired bus architecture for the XCC as it allowed reuse of the proven cores available from OpenCores.

4.2.5 Control Interface Data Protocol

The decision to select a point-to-point serial interface for the Control Interface necessitated a fairly complex gateware solution. The following functionality was required of the Control Interface data protocol:

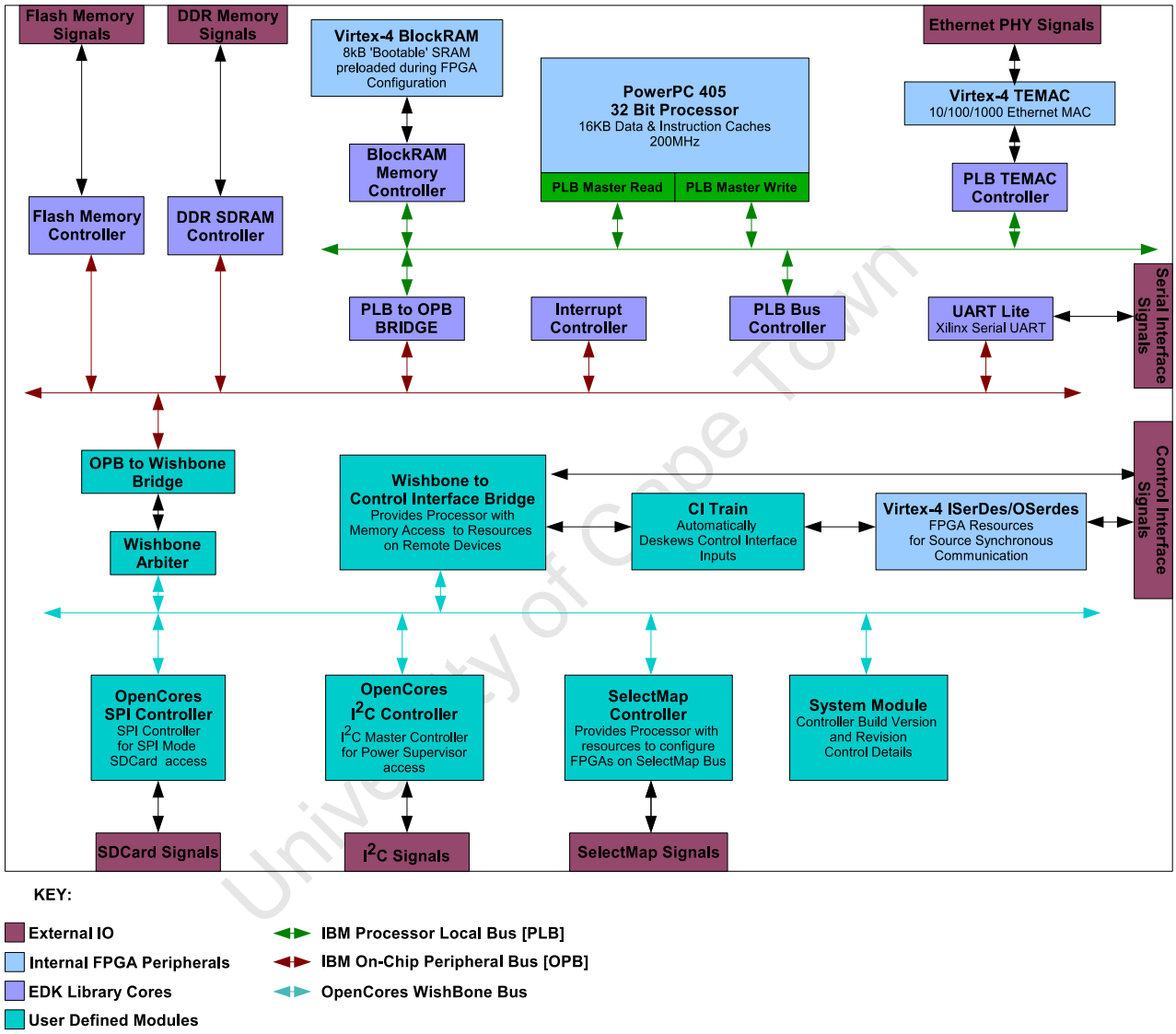


Figure 12: XCC Controller Gateware Block Diagram

- The control interface master required the slave memory to be mapped to its own memory.
- Reading from the slave should occur with high efficiency to support low bandwidth signal to be transferred from the XMCs to the control network.
- Slave devices should be able to issue interrupts to the master.

The control interface protocol was designed to be compatible with 4-bit SerDes devices. For this reason, the data protocol was based on transfers of 4-bit nibbles. These nibbles were transferred in sequences to form commands, which were initiated by the master, and responses, which were initiated by the slave. The following commands were implemented:

- IDLE: a single nibble command that indicated that the Control Interface was idle
- READ: a 5-nibble command that requested that a bus read operation on a 16-bit address be performed
- WRITE: a 16-nibble command that requested a bus write to the slave and included a 16-bit address, a 32-bit data word and a byte-enable signal to support byte addressing
- BULKREAD: a 7-nibble command that requested up to 256 reads from a single address on the slave and included a 16-bit address and 8-bit read count.
- BULKSWEEP: a 7-nibble command that requested up to 256 reads from a contiguous slave memory area and included a 16-bit address and an 8-bit read count.
- RESET: a single nibble command that issued a reset on the slave WishBone bus

The following slave responses were implemented:

- IDLE: a single nibble response that indicated that the Control Interface was idle
- READ: a 9-nibble response that returned the result of a read command and included a 32-bit data word
- WRITE: a single nibble response that indicated that a write command had been completed
- BULK: a variable length response that returned the data requested from a BULKREAD or BULKSWEEP command
- IRQ: a single nibble response that indicated the CI slave required attention

A typical read command experienced 32 cycles of latency[2], owing to the retiming introduced in crossing clock domains and the serialization and deserialization necessary for operating the serial control link. This equated to a 12.5% efficiency on the return data link. This would have been sufficient for control operations for which performance was not critical. However, this efficiency, while reading arrays of values out of slave elements, such as DSP structures, would most likely have proved insufficient. For this reason the bulk read transfers were implemented. These commands had an efficiency that could be calculated:

$$e = 8 * B/(34 + 8 * B)$$

where e was the response data efficiency and B was the bulk length. With the maximum bulk length of 256 transfers, the efficiency of the bus equalled 98.3%.

4.2.6 Source-Synchronous Interface Training

Virtex-4 FPGAs included resources called ISerDes and OSerDes, which were designed for serial source synchronous communication. They supported high data rates, allowing the FPGA fabric to operate up to 10 times more slowly than the serial I/O rate. Each ISerDes module included a 64-tap Delay-Locked Loop[DLL], which allowed the input data to be delayed up to 4.6 nanoseconds, and a bitslip sub-module, which implemented a barrel-shift on the input word. [10] These features enabled the user to perform training on the input data, allowing very narrow logic windows and, hence, facilitating higher data-rates. These interfaces could also negate the need for length matching across busses, as the time delay of the line could be compensated for using the delay elements.

The training scheme implemented on the Control Interface was largely based on the work presented in the Xilinx reference design: Dynamic Phase Alignment for Networking Applications.[11] When training was initiated, the transmitter outputted a fixed training word that was serialized, transmitted and then de-serialized by the receiver. The received parallel word was then used to establish bit-alignment and word-alignment. Bit-alignment refers to delaying the input data, so that the input buffer is sampled at what would be the most open point of the bit eye. Word alignment refers to shifting the bit-aligned training word, so that expected training word is received.

The bit-alignment operation is summarized in Figure 13. The receiver started training by incrementing the delay of the input serial data until the received parallel word changed to a shifted version of the training word. This check was required to ignore false values that occurred when the input data was sampled on the bit transitions. After a valid change had been detected, another change was sought by incrementing the delay. However, during this search, the delay increments were counted, as were the invalid words. Once the next valid change had been found, the delay was reversed by half the accumulated delay, less half the number of invalid words encountered during the search. This ensured

[2]Value establish using Verilog simulations

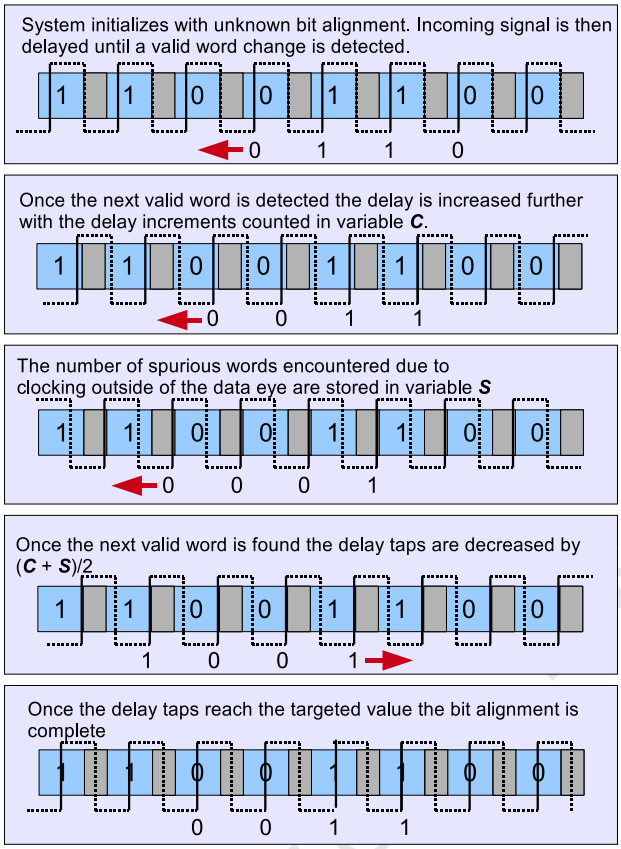


Figure 13: Control interface bit alignment

that the bit was sampled as close to the most stable point possible. In order to establish bit-alignment, the training word had to include at least one bit transition.

Once the input parallel word had been bit-aligned, word-alignment was implemented by shifting the input word, using the bitslip modules, until it matched the training word. In order for this operation to work, the N-bit training word had to be unique when compared to itself, circularly shifted by 1 up to N - 1, where N was the bit width of the training word.

4.2.7 SelectMap

The SelectMap interface was implemented as a very simple WishBone module. The module included registers that had their outputs directly tied to all SelectMap lines. This necessitated a bit-banging protocol in the software, involving manually toggling clock, data and control lines.

4.2.8 OpenCores SPI and I^{2}C Masters

Two OpenCores communications modules were used in the XCC Controller gateware: an SPI master controller and an I^2C master controller. The SPI module was intended to be used to access the SDCard in SPI mode. SPI mode was the slowest means of accessing SDCards, but was an attractive option, owing to its simplicity to implement. If additional performance had been required, a fully-featured SDCard controller would have been needed. The I^2C master was a well-established OpenCores module and supported 7- and 13-bit addressing, bus arbitration and interrupts.

4.2.9 Direct Memory Access Module

The efficiency of accessing registers on the PLB bus from the CPU was too low to operate the Ethernet link beyond 10 Mbps.[3] A Direct Memory Access [DMA] PLB bus master would have been required to achieve the full 1 Gbps. The Xilinx PLB Ethernet controller had the option to include a Scatter-Gather DMA engine in the controller itself. However, this option used almost a third of the FX12's slices, which made it impossible to fit all the required logic on to the device. A custom light-weight DMA engine would have needed to be developed to achieve performance on the PLB bus. Owing to time constraints, this was not achieved during the course of the project and is recommended for future work.

4.2.10 Pre-Integration Testing and Results

The XCC controller gateware was tested, using a development board called the Avnet Mini-Module Baseboard. This board included a USB serial port, which was used to establish communications between a host PC and the FPGA. Carabas, a bootloader developed by Marc Welz, was modified to include test routines that allowed the functionality of various modules to be established. The Control Interface was tested by looping back GPIO LVDS pairs on the development board to simulate real control interface data pairs. This allowed the control interface master, slave and training to be tested on the FPGA.

4.3 XCC Communications FPGAs

4.3.1 Design Overview

The interfaces into each communications FPGA are shown in Figure 14. XDM functionality required each Communication FPGA to transmit incoming data from the XDR attached to the XMC slot to

[3] Established by KAT members working on the Avnet module

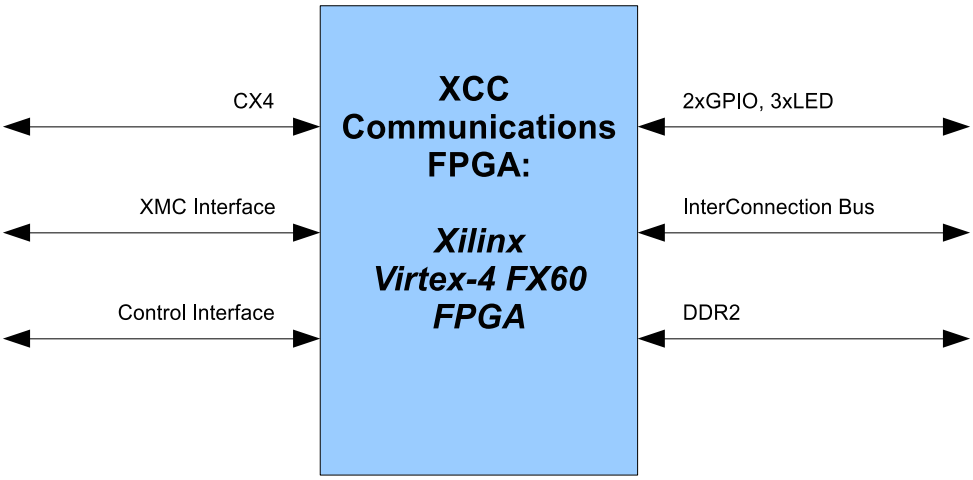


Figure 14: XCC Communications FPGA Interfaces

the 10 Gigabit Ethernet interface. However, at the time of the completion of this project, no XDR cards were available. As a result, the XMC interface, although fully developed, was untested and thus will not be presented. In its place, a dummy XMC module was developed to simulate data out of this interface and was included in the Communication FPGA gateware.

The design of the Communications FPGA gateware is shown in Figure 15. The control interface to WishBone bridge provided the controller with access to the registers of the WishBone slaves. Two types of Ten Gigabit communications modules were used: a XAUI controller and a 10 Gigabit Ethernet controller. The XAUI controller was a simpler module, which allowed only point-to-point communications, and was designed for testing purposes. The 10 Gigabit Ethernet controller generated true Ethernet and UDP packets and could be used through commercial network switches. The System Block was a basic module that performed miscellaneous tasks, such as storing revision information.

4.3.2 XAUI Pipe

Figure 16 illustrates the various functional layers of 10GBase-X, with reference to their equivalent layer in the OSI model. 10GBase-X is a set of Ten Gigabit Ethernet protocols that implements 8B/10B encoding. The communications may operate as either optical transmissions, as in 10GBase-LX, or electrical transmissions, as in 10GBase-CX4.[12]

The Physical Coding Sublayer [PCS] and Physical Media Attachment [PMA] layers were implemented on the Xilinx MGTs. The PCS layer implemented 8B/10B encoding, which was an encoding scheme that provided the following features: [13]

- Generation of DC-balanced outputs

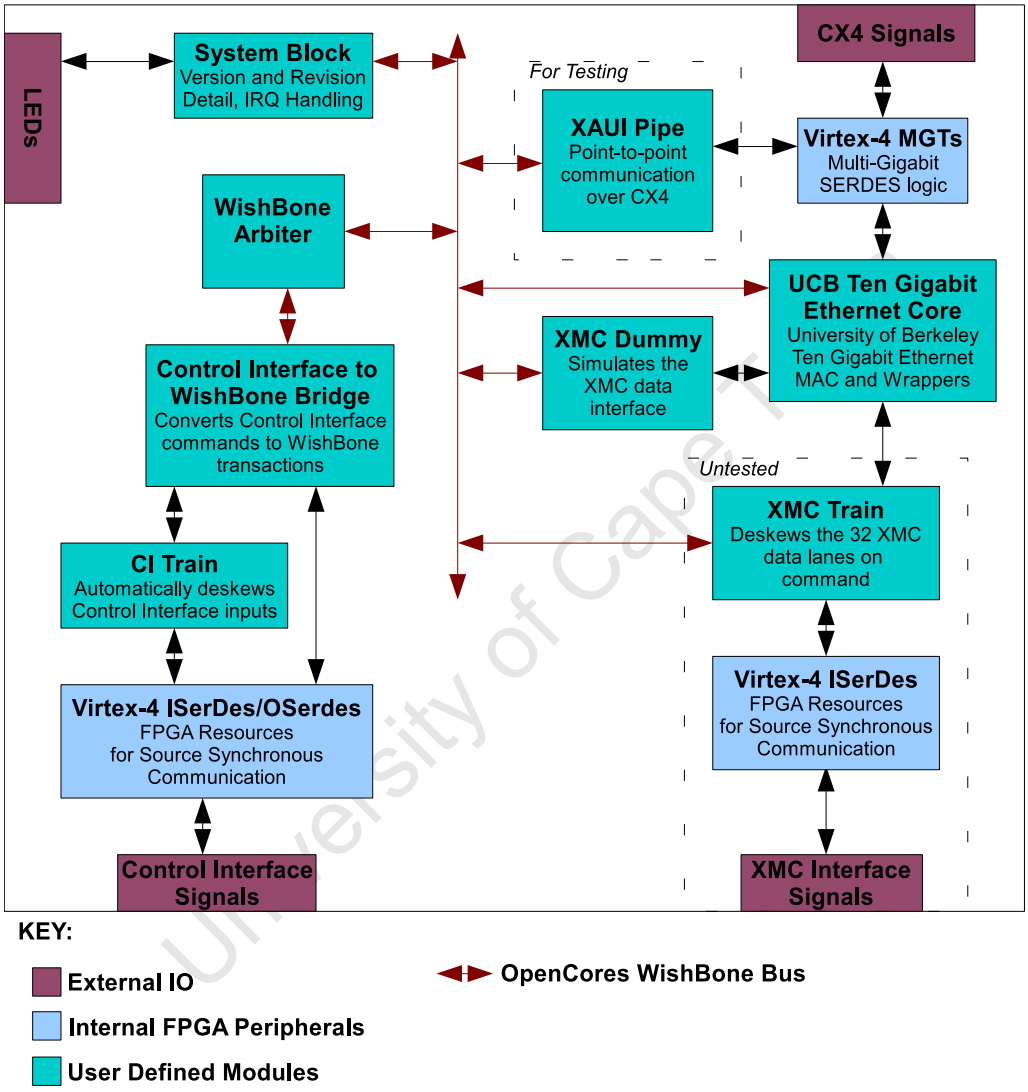


Figure 15: XCC Communications FPGA Gateware Block Diagram

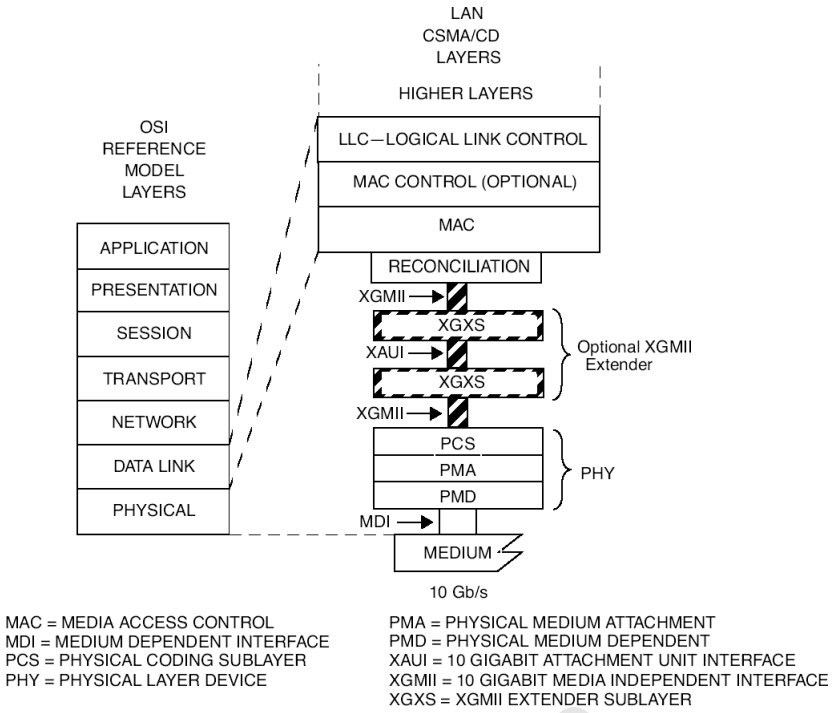


Figure 16: XAUI on the Ten Gigabit Ethernet OSI stack[12]

- Support of byte alignment
- Sufficient bit transitions to allow a receiver Phase-Locked Loop [PLL] to lock on to the incoming data source

The MGTs also supported clock correction and lane-to-lane synchronization.[8]

The function of the XAUI controller was to map incoming XGMII idle bytes into combinations of idle sequences that facilitated clock correction, lane-to-lane synchronization and byte alignment. In the design, this function was performed by a XAUI controller IP core provided by Xilinx. [12]

The XAUI Pipe module provided a simple 64-bit data read and write interface that chanelled data to the XAUI controller. It did not perform Cyclic Redundancy Checks [CRCs] and other features implemented by a MAC. This interface was suitable for simple tasks, such as those required for testing the performance of the Ten Gigabit Ethernet.

4.3.3 UCB Ten Gigabit Ethernet Controller

The signal processing and electronics research group of the University of Berkeley, California, CASPER, had released a Ten Gigabit Ethernet controller, which was adapted for this project. The controller was written in VHDL, included a MAC and used the Xilinx XAUI controller. It was designed for Xilinx

Virtex-II pro devices and had to be adapted to the Virtex-4 architecture. Furthermore, it included a PLB interface, which had to be converted to a WishBone interface.

The module had the following key features:

- Two data interfaces, one accessible through the bus attachment and one through the FPGA fabric
- Automated UDP packet generation through the fabric interface
- Address Resolution Protocol [ARP] support on the fabric interface

The Virtex-4 MGTs included dedicated silicon that implemented CRC generation and validating. This feature was not used in the adapted controller. However, if it had been utilized, a substantial saving in FPGA resources would have resulted. It is recommended it be used for future work.

4.4 Conclusion

This chapter described the gateware design for all four FPGAs on the XCC. A test-oriented approach to HDL development was used, with the goal of creating an automated verification and build system. Where possible, automated test infrastructure was developed for all HDL modules.

The XCC Power Supervisor gateware design included the following features:

- A power-sequencing engine
- Automatic power-downs for voltage, temperature or current threshold violations
- A 300 kb ring buffer of ADC samples for history analysis
- A flash memory backup of the ring buffer for crash analysis
- Serial Universal Asynchronous Receiver/Transmitter [UART] and I^2C slave interfaces for communications
- A conditional JTAG forwarding module for selectively distributing the JTAG chain

The gateware for the XCC Power Supervisor was tested on a development board.

The Xilinx Virtex-4 FX12 FPGA included a PowerPC processor that connects to the configurable FPGA fabric. The primary function of the XCC Controller gateware was to provide the software running on the processor with peripherals to perform control, monitoring and configuration tasks. The following gateware elements were attached to the memory map of the PowerPC:

- ROM, containing code programmed at FPGA configuration time
- DDR SDRAM and flash memory controllers
- OpenCores I^2C and SPI controllers
- A High-speed control interface master
- A SelectMAP controller for programming FPGAs

The XCC Controller gateware was tested, using an Avnet FX12 Mini-Module Baseboard. The communications FPGA included three modules:

- A high performance, source synchronous control interface slave controller
- A ten Gigabit Ethernet controller with a WishBone interface
- A simulated XMC interface, connected to the ten Gigabit Ethernet module

5 System Verification and Performance

This chapter described the testing procedures and performance of the XCC hardware and gateware.

5.1 Bringing up the board

5.1.1 Power up tests

The first powering up of the board was executed very carefully to prevent possible damage to the components, owing to manufacturing or design error. Thus, several tests were performed without the power switched on.

The first test carried out off-line was the measurement of the impedance between ground and power supply traces. The idea behind this test was to isolate and fix short circuits that might have occurred. A multimeter was used to perform these tests and no short-circuits were found.

The next test involved checking the power supply connectors. Owing to the absence of a Compact PXI Express backplane during testing, the ATX power supply was the only power connector tested. It was found that the pin-ordering on the connector was incorrect. This problem was circumvented by modifying a standard ATX supply cable to match the incorrect pin assignments on the ATX connector.

5.1.2 Auxiliary Power

The power management circuitry ran off an always-on 5V auxiliary power supply. This circuitry was tested, using a current-limited power supply. Once the auxiliary supply had been switched on, the two regulated auxiliary voltages were measured and were found to be correct.

Once it had been established that the power supply for the Actel Fusion FPGA was functioning, the device was configured, using JTAG, with a LED flashing test. This worked as expected. The Power-On Reset [POR] circuit for the Fusion device was also tested. The reset was generated from the output of the Fusion FPGAGOOD signal. Figure 17 shows the measured response of the POR. This response generated a reliable reset condition after power-up.

5.1.3 Full Power

Once the Actel Fusion FPGA was working, the FET gate drivers were used to switch on the full power. Trouble was encountered with switching on the ATX power supply. The supply would shut itself down if the 3.3V power was unloaded when powered on. This was most likely due to minimum power requirements on the supply. A simple circumvention solved this problem: the 3.3V supply was switched on after the 12V and 5V supplies. The precise cause of the problem was not investigated, as

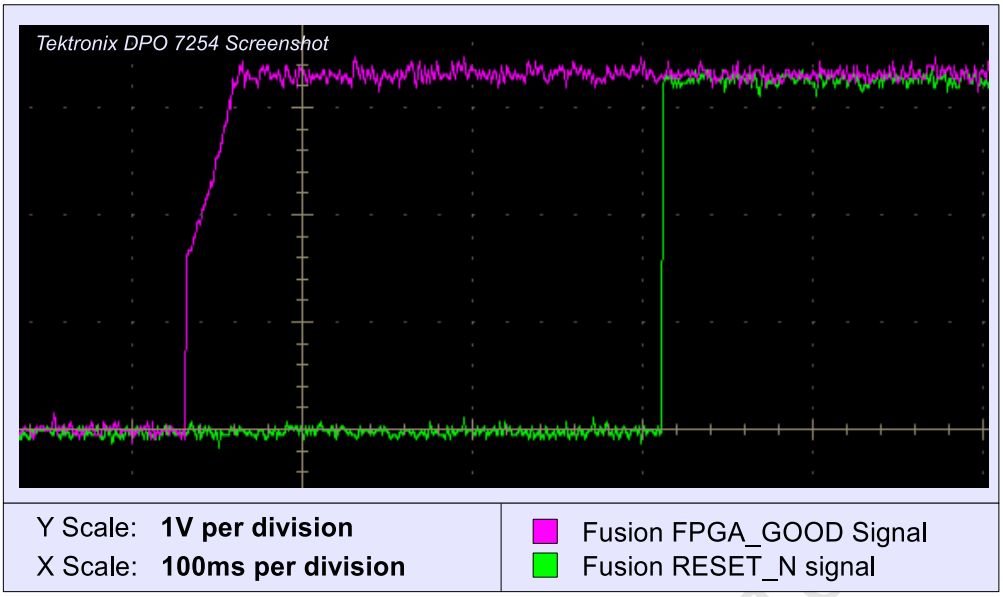


Figure 17: Actel Fusion Power-On Reset

the ATX power was for bench operation only. The VTrack signal to the 2.5V power supply had been incorrectly passed through a FET switch and this caused the power supply chain to fail. The FET was bypassed with a wire and the problem was resolved.

The measured power supply voltages are summarized in Table 5.1.3. All supplies listed in the table met the requirements specified by their target devices. However, there was a problem with the DDR2 termination voltage supply. This device had its feedback connection incorrectly tied to the 0.9V reference voltage, rather than to the termination voltage and this caused the termination output to hit the 2.5V supply rail. This problem could have been resolved by severing the offending trace and using a wire to connect the termination voltage to the feedback input. This was not done as the DDR2 memory was not tested.

5.2 Power Supervisor Testing

5.2.1 Serial Communications

A loopback test was performed in order to establish the reliability of the serial port. During this test, the two GPIO pins connected to the Fusion FPGA were attached to a serial port on a PC, using an external RS232 level conversion circuit to convert the LVTTL voltage levels to those compatible with

Table 3: Power supply performance summary

Supply Name	Measured Voltage [V]	Device Recommended Range [V]
3V3 AUX	3.290	2.97 to 3.63 [a]
1V5 AUX	1.560	1.425 to 1.575 [a]
2V5	2.518	2.375 to 2.625 [b]
1V8	1.797	1.7 to 1.9 [c]
1V5	1.489	1.14 to 1.575 [b]
1V2	1.197	1.14 to 1.26 [b]
1V2 MGT 0	1.204	1.14 to 1.26 [b]
1V2 MGT 1	1.207	1.14 to 1.26 [b]

[a] Actel Fusion Datasheet
[b] Xilinx Virtex-4 Datasheet
[c] JEDEC DDR2 SDRAM Specification

RS232. A megabyte of random values was transferred at a rate of 115200 BAUD from the PC to the board, which returned the data to the PC, where it was stored and compared to the original data. It was found that no errors had occurred.

The next test involved programming the Actel with the full gateware as described in Chapter 4. The communications into the final gateware over the serial port worked reasonably well, very occasionally losing commands. This minor problem was not investigated, as the serial port was for testing purposes only. I^2C was the primary mode of communication between the controller and the XCC Power Supervisor and this worked without error. [This indicated that the lost commands in the serial port mode were due to the UART used.]

5.2.2 Analogue Block Performance

The performance of the voltage monitoring of the analogue block logic is summarized in Table 5.2.2. The errors presented in the table represent the variation from values measured using a multimeter. The variable offset errors can probably be attributed to the Fusion prescaler circuitry. The accuracy achieved would be adequate for most applications. However, if further accuracy were required, a calibration scheme should be implemented to minimize these errors. The Actel Fusion Handbook describes in detail how this can be achieved.[7]

The current and temperature measurement were implemented incorrectly in the XCC Power Supervisor gateware. This was due to the incorrect handling of the current and temperature strobes in the Fusion analogue block. The solution to this problem would require a redesign of the ADC controller and is recommended for future work.

Table 4: Fusion Voltage Monitoring Performance

Supply Name	Measured Voltage [V]	Error [%]
5V AUX	5.017	0.26
1V5 AUX	1.439	3.5
12V	11.89	1.86
5V	4.967	2.71
3V3	3.292	0.77
2V5	2.503	0.60
1V8	1.768	1.63
1V5	1.439	3.5
1V2	1.151	4.02

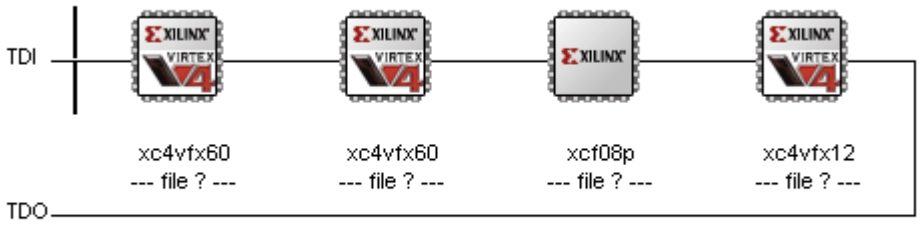


Figure 18: XCC JTAG chain using Fusion JTAG forwarding as represented by Xilinx Impact tool

5.2.3 Board JTAG

The board JTAG for programming the Xilinx devices on the XCC was then tested. The board JTAG connector had been soldered on to the board backwards, owing to an ambiguous symbol on the PCB silkscreen. The manufacturers should be notified of this problem should further boards be assembled.

The JTAG pins of the controller modules had been incorrectly assigned on the schematic, but this caused no problems, as the I/O allocation on the Fusion device could be adapted in compensation. Figure 18 illustrates the JTAG chain as represented by Xilinx's Impact software. It shows both FX60s, the FX12 and the controller module PROM.

5.3 Controller Module Testing

The serial port used the same external RS232 circuitry as the Fusion device. This communication mechanism between host PC and the software running on the FX12 worked without error. The external RS232 circuitry could have been included on the board, as a serial console is a useful means of communication. The I^2C master gateware and software worked without error.

5.4 Communications FPGA Testing

JTAG programming of the Communications FPGAs worked as expected. The SelectMap configuration interface was untested, owing to the software nature of the problem.

There was a problem with the 200 MHz clock generation circuit. The LVDS output on one of the pairs of the Texas Instruments clock multiplier demonstrated a DC bias in its voltage swing. The cause of this problem was not investigated as the control interface clock could be used in place of the 200 MHz clock.

5.5 Control Interface

The interface was tested by issuing 10^8 write and read commands. This amounted to a total of 10.6 Gb of data without error, a BER of less than 10^{-10}.

The test took 520 seconds to complete, which equated to a performance of 20 Mbps. This performance was far lower than the 157 Mbps expected and could have been influenced by the following:

- Latency in the SerDes
- Poor Bus performance, owing to the PLB to WishBone bridge
- Software test routine overhead

The likely cause of the poor performance was high bus overheads. This problem could be alleviated by introducing a DMA module or by optimizing the PLB to WishBone bridge.

5.6 Ten Gigabit Ethernet Testing

5.6.1 Eye Diagram Measurement

During this test, one of the FX60 FPGAs was configured to transmit a counter over the Ten Gigabit Ethernet links, using the XAUI Pipe module. The signals were transmitted over both 1m and 5m cables. A 8 GHz differential probe was soldered to a modified CX4 connector, which connected to the receiving end of the cable. A Tektronix DSA 70804 digital serial analyzer was used to analyze the incoming digital signal.

Figure 19 shows the eye diagram, as measured by the digital serial analyzer over both 1m and 5m cables. The results indicated that 5m was close to the maximum cable length that could be utilized.

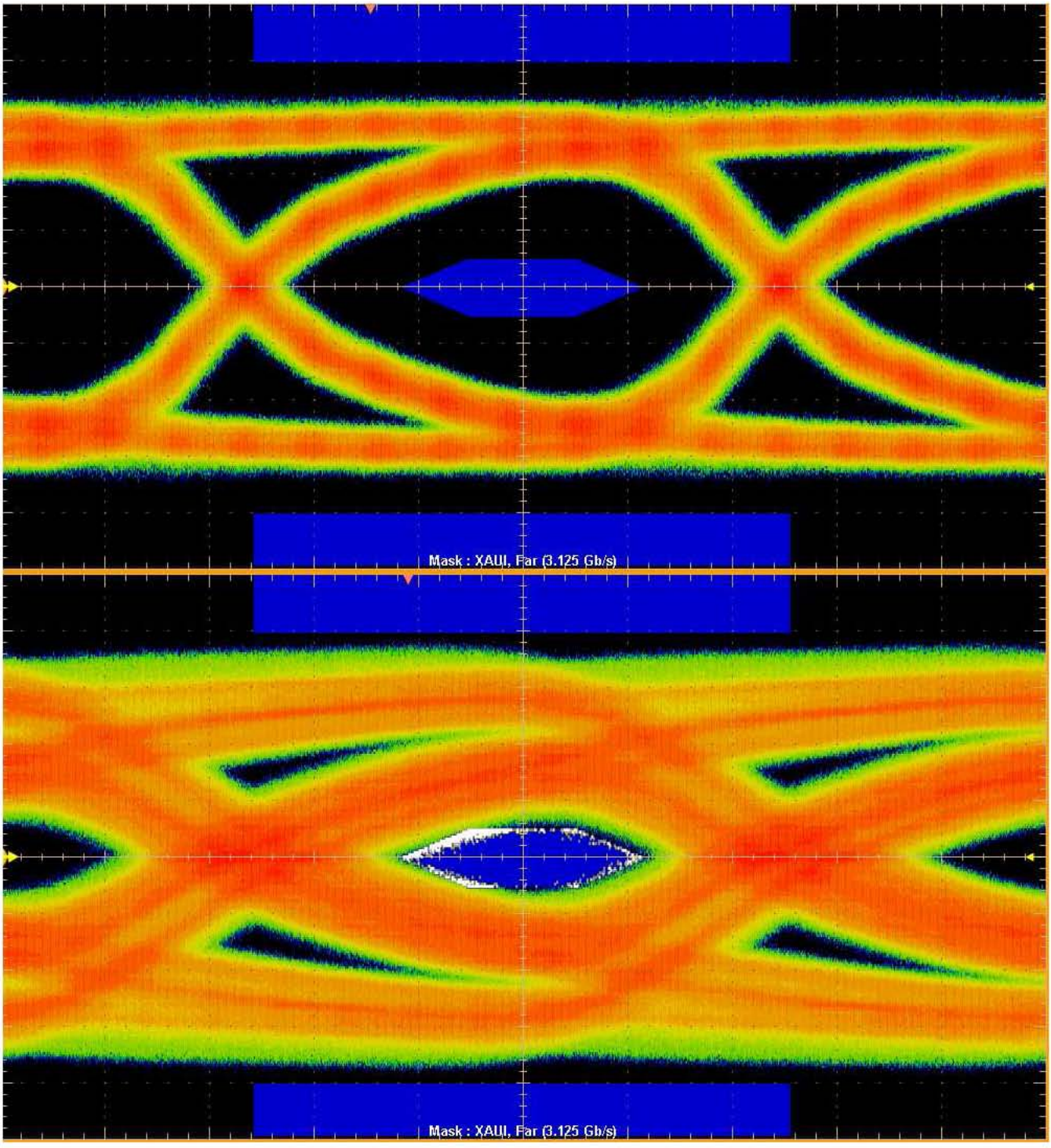


Figure 19: Ten Gigabit Ethernet eye diagram achieved over 1m (top) and 5m (bottom) cables

5.6.2 Bit Error Rate Testing

The BER test was performed by connecting a 5m CX4 cable between the two Ten Gigabit Ethernet ports on the XCC. The XAUI Pipe module was used to transmit a counter at close to 10 Gbps. This test showed no error when transmitting more than 10 terabits of data, a BER of less than 10^{-13}.

5.6.3 Communications with a Myricom Ten Gigabit Ethernet Card

A test was necessary to validate that the Ten Gigabit Ethernet on the XCC could operate correctly when communicating with commercial Ten Gigabit Ethernet technology. This was done by using the UCB Ten Gigabit Ethernet controller to transmit UDP packets to a Myricom Ten Gigabit Ethernet PCI Express network adapter, which was plugged into a Dell PowerEdge 1950 server.

The network performance was measured using a tool called *iptraf* on a Debian Linux system. The tool reported a maximum sustained data rate of 7.5 Gbps, which was limited by the performance of the server.

5.7 Electromagnetic Radiation Testing

The Compact PCI architecture specifies the use of mezzanine cards that plug in above their host cards. This can lead to increased EM radiation pickup on the daughter cards from the base card. For this reason, the radiation emitted from elements on the XCC was investigated.

The radiation was measured using a 15cm dipole antenna, located 10cm away from the source under investigation. The analogue signal was sampled and analysed, using a Tektronix DPO 7254 oscilloscope. It was found that the primary source of radiation was the Avnet controller module. Figure 20 shows the spectral power of the measured radiation with the device both off and on. The EM radiation from the controller module caused signals to be received of more than 20dB from the noise floor. It is likely that this radiation would be picked-up by an ADC on the mezzanine card. The full extent of this problem should be investigated, using data recording from an attached ADC module.

5.8 Thermal Observations

Low-profile heatsinks were connected to the FX12 and both FX60 FPGAs during testing. Without airflow, the FX12 became too hot to touch and the two FX60s became slightly warmer than room temperature. A small PC power-supply fan was used to blow air over the heatsinks, which lowered their temperatures to room temperature.

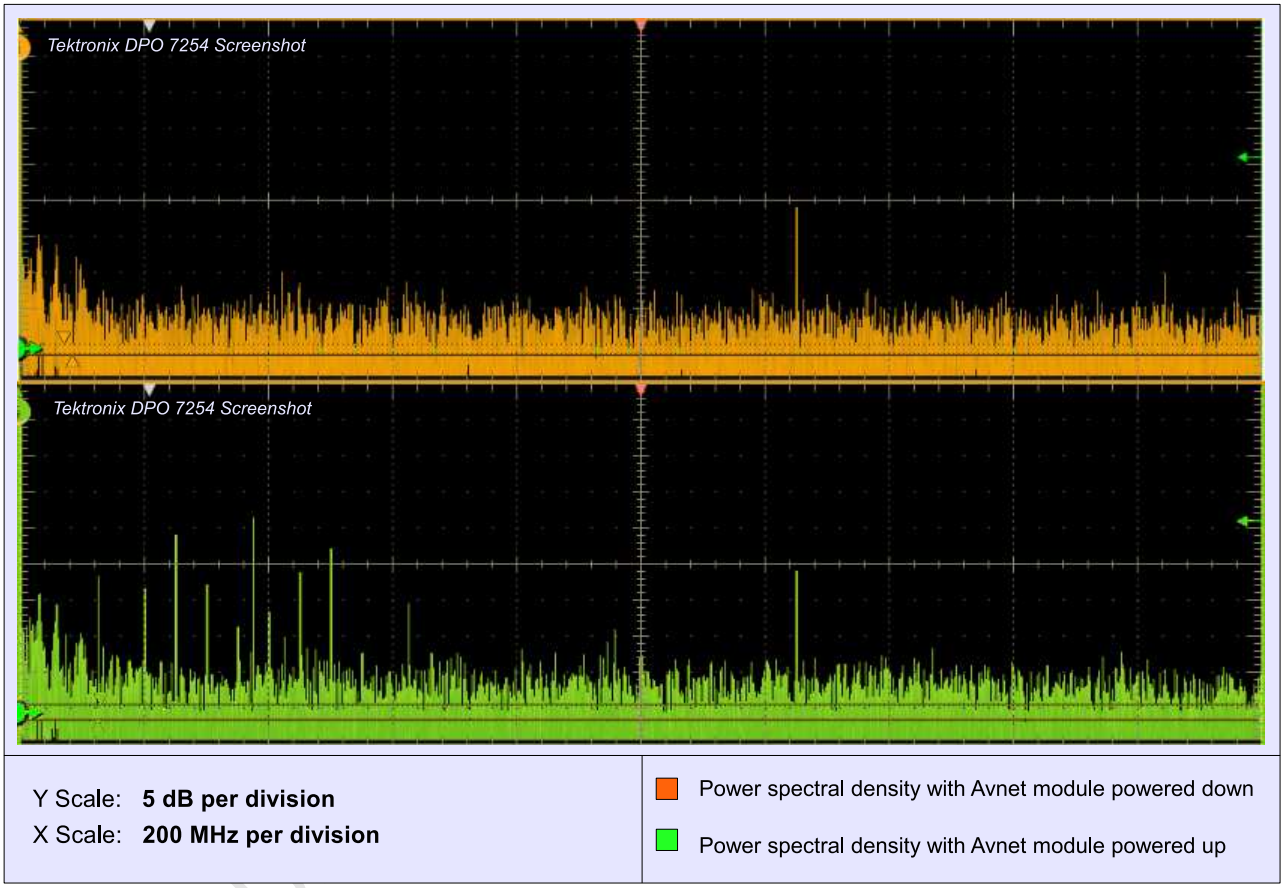


Figure 20: Power Spectral Density of EM radiation from Avnet FX12 Module

Given that the FX60 FPGAs were only 15% utilized and that the mezzanine cards would both generate heat and obstruct airflow, it is certain that further investigation into the cooling requirements of the final system would be necessary.

5.9 Conclusion

This chapter describes the testing and verification procedures and reports on the board performance.

The first tests were carried out with the power off-line. These included short circuit and power supply connector pinout checks. It became evident that the bench ATX power supply connector had the incorrect pin-ordering. A standard ATX power supply was modified to overcome this problem.

The next set of tests were carried out on current-limited auxiliary power. All three supplies derived from the auxiliary power worked as anticipated. The Actel Fusion FPGA was then programmed with a LED flasher gateware image, and this worked as anticipated. The Power-on Reset [POR] was checked and found to reliably reset the FPGA.

Next, the full power was tested. The gate drivers of the Actel Fusion device successfully switched on the power from the ATX supply. All power supplies met the required specifications, with the exception of the DDR2 termination supply, which, owing to a schematic error, failed to work. This supply was left disabled, as the DDR2 memory was not required for XDM.

The XCC Power Supervisor was tested, using a serial port that was connected to GPIO via an external RS232 level converter. There were minor reliability problems with the serial UART. These were not investigated, because the serial port was intended for debugging purposes only.

The accuracy of the Actel ADC was then tested. The performance was measured by comparing the voltages obtained from the ADC to those measured by a multimeter. The ADC measurements had between 0.2 and 4% fixed offset error, and calibration would be required for improved accuracy. The current and temperature monitoring was implemented incorrectly and did not work. The solving of this problem would involve a redesign of the XCC Power Supervisor and is recommended for future work. The JTAG forwarding worked once the incorrect pin allocations of the Avnet Module had been compensated for: Each of the FPGAs on the JTAG chain could be programmed.

The XCC Controller was tested, using embedded software running on the PowerPC. Communications with the software were established, using the same RS232 level converter system as had been used previously. No errors were observed on this serial port. The I^2C connection between the XCC Controller and the XCC Power Supervisor worked reliably, with no errors encountered. The control interface was tested and showed no error when transferring ten gigabits of data. This equates to a Bit Error Ratio [BER] of less than 10^{-10}.

Eye diagrams for the Ten Gigabit Ethernet were measured using a Tektronix DSA 70804 digital serial analyzer and were sufficiently open at cable lengths of 5 metres. The next test involved transmitting

a counter from one Ten Gigabit Ethernet link to the other and checking that the output remained a glitch-free counter. No errors were encountered over the 5m cable when transmitting over ten terabits of data, which amounted to a BER of less than 10^{-13}. The final Ten Gigabit Ethernet test involved using a Ten Gigabit Ethernet core to transmit UDP packets to a Myricom network adaptor, plugged into a Dell server. The link was found to operate up to 7.5 Gbps and was limited by the performance of the server.

6 Conclusions and Further Work

This chapter presents the conclusions of the project based on the results and performance achieved of the designed system. Of particular interest is to what extent the project requirements, as set out in Chapter 2, were met. Also presented in this chapter are recommendations for further work, which will either improve the performance of the board or will test elements on the board that were untested at the conclusion of this project. This section also presents design ideas for future revisions of the board.

6.1 Conclusions

The primary goal of this project was to develop a hardware platform that, when integrated into the XDM DBE subsystem, would satisfy all relevant requirements. It was shown during the testing and verification phase that this was achieved. The board performed all the necessary functions for XDM after a number of minor hardware modifications were made.

The goal of mitigating risk for future KAT design iterations was also achieved. This was the first hardware project completed by the DBE subsystem and provided an opportunity to understand the risks associated with designing and manufacturing high-speed digital hardware. Also, several technologies that would be used in future DBE designs were investigated and shown to work. These included technology such as Ten Gigabit Ethernet and Actel Fusion FPGA solutions.

Unfortunately, two central hardware elements were unavailable at the time of the completion of this project: an XMC slave card and a PXI backplane. This meant that two functional requirements of the system could not be met. The XMC interface, although untested, was electrically similar to the control interface. Thus, it was presumed that this interface would function optimally. However, the backplane-related circuitry performance remained uncertain.

The scope of this project included the development of hardware, gateware and test software to verify the XCC performance and to achieve XDM functionality. Several gateware and hardware elements were left untested, owing to time constraints. The incomplete tasks are summarized in the Further Work section.

The system included several schematic, layout and gateware errors. All the problems encountered are summarized in Appendix C. Included in this summary are the proposed or implemented solutions.

6.2 Further Work

6.2.1 Untested Hardware

The following hardware was untested:

- Backplane power and power management
- Backplane clock and synchronization buffering and distribution
- Various XMC related items
- DDR2 Memory
- miniSD Card connection

6.2.2 Untested Gateware

The following gateware had been developed but was untested:

- OpenCores SPI controller
- SelectMap FPGA configuration controller
- XMC interface controller

6.2.3 Performance Optimization Tasks

The following tasks are required to improve the performance of the board:

- A light-weight DMA module would greatly improve the performance of software on the XCC Controller.
- The Ten Gigabit Ethernet Controller should be modified to use the CRC logic on the MGTs.

6.3 Design Improvement

Should a revision of the XCC occur, the following design improvements are recommended:

- The 200 MHz clock generation for the FX60s should be simplified by using a 200 MHz clock oscillator in place of the 25 MHz oscillator and mixer.
- An RS232 voltage level conversion chip should be included on the board to support serial ports for the controller and the power supervisor, thus negating the need for external circuitry.
- The unused MGTs on the Virtex-4 should be used to support an additional four CX4 ports. This would require a ATCA style rear transition module to connect to the back of the card, to provide space for the connectors. This card should also include a CX4 retimer chips if support for 15 metre cables is required.

References

[1] VITA: *Auxilliary Standard for XMC Switched Mezzanine Card*, http://www.vita.com, September 2005

[2] Lattice Semiconductor Inc.: *LatticeSC-M Data Sheet*, http://www.latticesemi.com/dynamic/view_document.cfm?document_id=19028, January 2008

[3] Altera Inc.: *Altera Stratix-II GX Handbook*, http://www.altera.com/literature/hb/stx2gx/stxiigx_handbook.pdf, October 2007

[4] Xilinx Inc.: *Virtex-4 Data Sheet*, http://www.xilinx.com/support/documentation/data_sheets/ds302.pdf, October 6 2006

[5] Memec Inc.: *Virtex-4 FX12 Mini-Module User Guide*, http://www.em.avnet.com, October 2005

[6] Actel Inc.: *Actel Fusion Data Sheet*, http://www.actel.com/documents/Fusion_DS.pdf, April 2006

[7] Actel Inc.: *Actel Fusion Handbook*, http://www.actel.com/documents/Fusion_HB.pdf, April 2006

[8] Xilinx Inc.: *Virtex-4 RocketIO Multi-gigabit Tranceiver User Guide*, http://www.xilinx.com/support/documentation/user_guides/ug076.pdf, August 17 2007

[9] Howard Johnson and Martin Graham: *High-Speed Digital Design: A Handbook of Black Magic*, Prentice-Hall, Inc., 1993

[10] Xilinx Inc.: *Virtex-4 User Guide*, http://www.xilinx.com/support/documentation/user_guides/ug070.pdf, April 10 2007

[11] Xilinx Inc.:: *Dynamic Phase Alignment for Networking Applications*, http://japan.xilinx.com/support/documentation/application_notes/xapp700.pdf

[12] IEEE Computer Society: *IEEE Standard 802 Part 3 2005: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and physical layer specifications*, http://standards.ieee.org/getieee802/802.3.html

[13] A. X. Widmer and P. A. Franaszek: *A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code*, http://www.research.ibm.com/journal/rd/275/ibmrd2705D.pdf

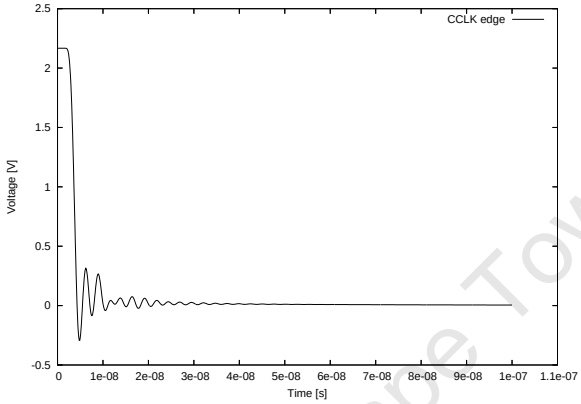
Appendix A: Contents of Attached CD

The attached CDROM includes material relevant to this thesis that could not be presented in the appendices. This includes schematics, layout files, manufacturing information and source-codes.

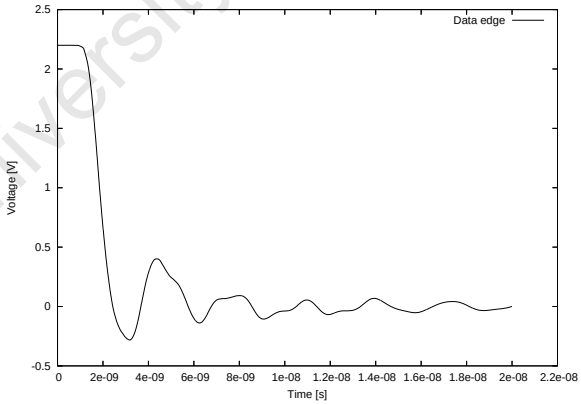
Appendix B: HyperLynx Simulation Results

SelectMap

The SelectMap bus connected the controller to FPGAs on the XCC and on XMC cards. The post-layout, simulated performance of the buffered, thevenin termination clock line, using the slowest Xilinx Virtex-4 slew rate:

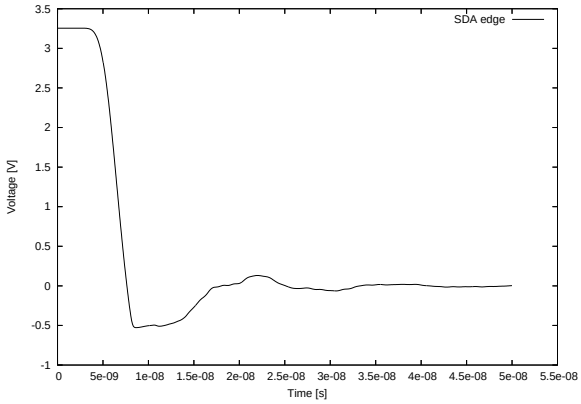


The post-layout, simulated performance of the unbuffered data line, using the slowest Xilinx Virtex-4 slew rate:



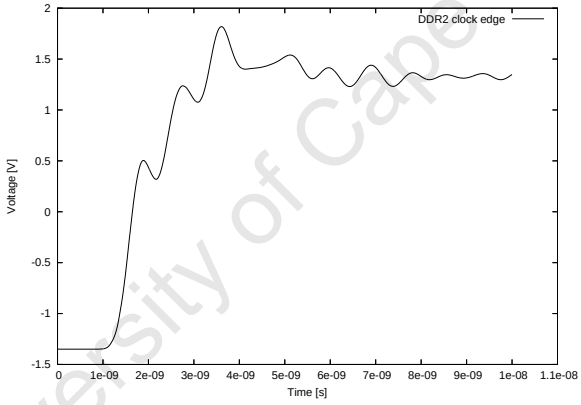
I^2C

The I^2C interface connected the controller to power management elements on the XCC and XMC cards. The post-layout, simulated performance of the unbuffered SDA line, using the slowest Xilinx Virtex-4 slew rate:

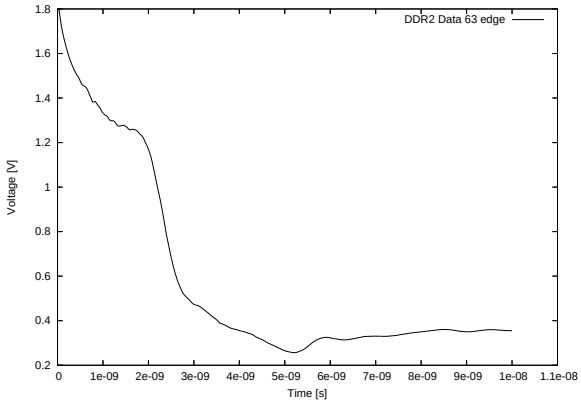


DDR2 Memory

The DDR2 memory interface connected the FX60s to a single DDR2 SODIMM. The post-layout, simulated performance of the worst clock line, using a Micron DIMM simulation model:

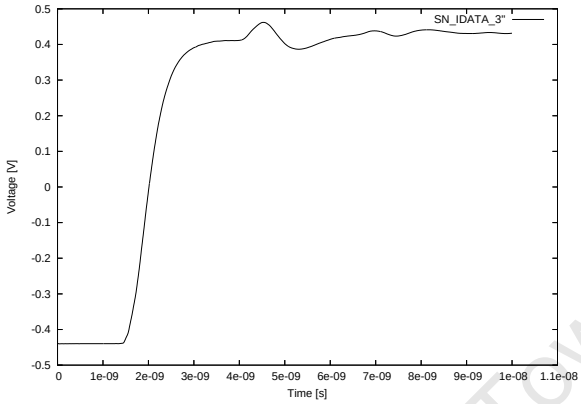


The post-layout, simulated performance of the worst data line, using a Micron DIMM simulation model:



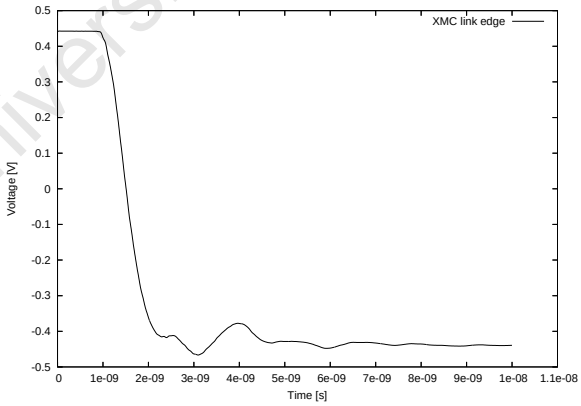
Control Interface

The Control Interface was a point-to-point, LVDS interface between the controller and FPGAs on the XCC and XMC cards. The post-layout, simulated performance of the longest line, including the routing on the XDR:



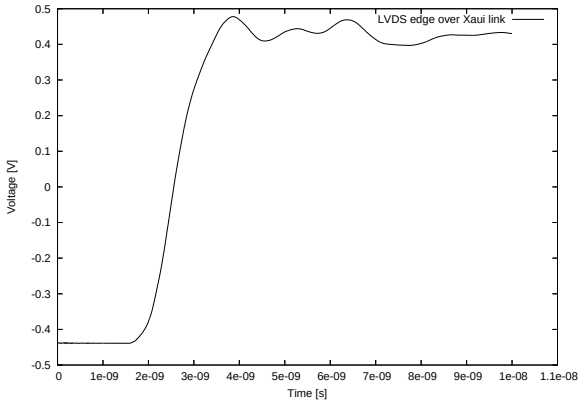
XMC

The XMC Interface was a point-to-point, LVDS interface between the FX60 FPGAs and the FPGAs on the XMC cards. The post-layout, simulated performance of the longest line, including the routing on the XDR:



XAUI

The XAUI interface was a point-to-point CML interface that connected to other boards, using a CX4 link. The post-layout, simulated performance of the longest line, connecting to a 15m transmission line and back to the XAUI port on the XMC:



This result was obtained using a Virtex-4 LVDS simulation model, as no MGT model was available.

Appendix C: Board Errata

Schematic

- The DDR2 SDRAM termination voltage generation circuit connected the VTTSNS signal to the VTTREF signal when it should have been connected to the VTT signal. This problem could be solved by severing the connection between VTTSNS and VTTREF and then soldering on a wire between VTTSNS and VTT.
- The FET switch on the VTrack signal was incorrectly added on the 2V5 supply generation circuit. This problem was solved by removing the offending FET and soldering a wire between the VTrack control line and the VTrack on the 2V5 power supply.
- The pull-up on the chassis switch was incorrectly pulled-up to 3V3 volts, rather than 3V3 auxiliary. Using an internal weak pull-up on the Fusion produced the intended behaviour.
- The JTAG pins on the Avnet Mini-Module were incorrectly assigned. However, as those signals connected directly to the Fusion FPGA they could be compensated for accordingly.
- The Control Interface outputs were connected to sites on the FX60 FPGA that did not support LVDS outputs. The problem was resolved by swapping the direction of the Control Interface lines on both the FX12 and FX60s.
- The 200 MHz clock generation circuit failed to work, owing to an unresolved issue with the clock multiplier. This circuit was overly complicated and a single 200 MHz clock oscillator should have been used.
- The XCC Controller and XCC Power Supervisor required external RS232 level-conversion circuitry to operate a serial port. A level conversion IC should have been included on the XCC for these interfaces.

Layout

- The ATX power connector had the incorrect PCB symbol. An ATX power supply cable was modified to support this connector during testing. It is recommended that the backplane power be used to avoid this problem.
- Both JTAG connectors had the notch on the silkscreen symbols oriented opposite to the notch on the connector. Manufacturers should be made aware of this problem to avoid the connectors being soldered on incorrectly.

Gateware

- Final XCC Power Supervisor gateware exhibited occasional loss of commands when communicating over the serial port. The cause of this problem was unknown. However, the I^2C interface did not lose commands, which indicated a problem with either the UART or the serial communications with the PC.
- The temperature and current monitoring on the XCC Power Supervisor did not work, owing to the current and temperature strobes being handled incorrectly. To solve this problem, the ADC controller module would need to modified to manage these strobes.
- Bus performance on the XCC Controller gateware was too low for high data-rate tasks, such as one gigabit Ethernet and Control Interface operation. A simple DMA controller would be required to improve this performance.
- The Ten Gigabit Ethernet controller did not utilize the CRC logic on the MGTs. The module should be altered to use this feature. This would save approximately 5% of slice resources on the FPGA.

Untested Elements

- DDR2 memory hardware
- Backplane power, synchronization, timing and power management
- XMC gateware and hardware
- SDCard hardware
- SPI controller gateware
- SelectMap gateware
- Thermal performance with integrated heatsink