To the Graduate Council:

I am submitting herewith a thesis written by Joshua Brandon Jones entitled "FPGA Logic Design for Analog-to-Digital-Converter Hardware Utilizing High Speed Serial Data Links". I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Bruce W. Bomar
Major Professor

We have read this thesis and
recommend its acceptance:

L. Montgomery Smith

Bruce W. Whitehead

Acceptance for the Council:

Linda Painter
Interim Dean of Graduate Studies

(Original signatures are on file with official student records.)

FPGA Logic Design for Analog-to-Digital-Converter Hardware Utilizing High Speed Serial Data Links

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Joshua Brandon Jones
December 2006

ACKNOWLEDGEMENTS

The author wishes to thank Dr. Bruce Bomar for his advice and assistance in the preparation of this thesis. Appreciation is also expressed to the University of Tennessee Space Institute, the United States Air Force (USAF), the Arnold Engineering Development Center (AEDC), and the management of Aerospace Testing Alliance for the opportunity to pursue this graduate degree and the resources required to complete this masters thesis.

The author also wishes to thank Mrs. Sharron Cockrell, Dr. Gary Cockrell, Mr. Jessie Jones and Mrs. Rita Jones for their support, advice and assistance throughout this study. The author would also to extend his deepest gratitude to Mrs. Jennifer Jones for her patience and support throughout the authors' entire Masters' education.

The research and results reported herein was performed by personnel of Aerospace Testing Alliance, support contractor for AEDC. Approved for Public Release; Distribution is unlimited. Further reproduction is authorized to satisfy the needs of the U.S. Government.

ABSTRACT

The Computer Assisted Dynamic Data Monitoring and Analysis System (CADDMAS) used by the aeropropulsion test cells at Arnold Engineering Development Center (AEDC) processes a large amount of high bandwidth accelerometer and strain gage data. Data from each sensor must be digitized before being processed by software running on networked computers. This thesis describes some of the original analog-to-digital converter (ADC) hardware used by the CADDMAS as well as some of its limitations. More up-to-date ADC hardware was designed to be used in the CADDMAS to enhance capabilities required by the aeropropulsion test cells. These new capabilities included an increase in maximum sample rate for the CADDMAS, a Universal Serial Bus (USB) 2.0 interface for easy connection to the computer, the latest Field Programmable Gate Arrays (FPGAs) for controlling the data acquisition, and a packet based data transmission scheme to reduce redundant data transfers into the computer. This thesis describes, in detail, the development and checkout of the FPGA logic for the new ADC hardware needed to obtain these enhanced capabilities. A simple software program was also developed to validate the correct operation of the new ADC hardware and to demonstrate to a commercial software vendor how to incorporate support for the ADC hardware into their software. Upon completion of the designs, a maximum data transfer rate of 96.15 Megabytes per second (MBps) was obtained using a PCI interface card and 18.45 MBps was obtained from the USB 2.0 interface.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Arnold Engineering Development Center (AEDC) houses one of the largest collections of flight simulation test facilities in the world. AEDC is a unit of the Air Force Materiel Command and is a test and evaluation center for the United States Air Force and the Department of Defense. Engineers at AEDC have been involved in the development of nearly all U.S. military high performance aircraft as well as many NASA space systems.

Testing at AEDC is divided into three major categories: aerodynamics, aeropropulsion, and space and missiles. Aerodynamic testing includes the measurement of performance, stability, control systems, and aerodynamic load of an aircraft or other aerodynamic model. Aeropropulsion testing concentrates on the aircraft propulsion system and on the performance characteristics of these engines under varied flight-test conditions. Finally, the Space and Missiles department evaluates the performance of entire rocket systems, from the engine to the flight controls. This department also includes resources for evaluating system performance in the vacuum and hard radiation environments of the type expected in space.

The majority of the aeropropulsion testing involves measurements of vibrations and pressures with bandwidths in excess of 5,000 Hertz (Hz). Large channel counts of several hundred are also typically present during a test. For each channel present, all of these measurements must be synchronously acquired, processed, displayed, and stored in near real-time. This requirement places strict specifications on the hardware and software used. The current system used to provide this capability is the Computer

Assisted Dynamic Data Monitoring and Analysis System (CADDMAS).

CADDMAS is a networked, distributed processing system capable of hosting multiple front-end processing (FEP) units and multiple display stations. The first step in processing each channel involves converting the analog signal to the digital domain. This allows common commercially available personal computers (PCs) to act as FEPs and to provide further processing, displaying, and storage of the data. This is accomplished with the analog-to-digital converter (ADC) hardware. Once the analog signals have been converted into a digital data stream, it is collected, processed, and distributed via an Ethernet network by the FEPs. Each display station, also a PC, receives the processed data over its Ethernet link and displays it to the user. The display stations can also store the data at the discretion of the user. The software used to accomplish all of the processing, displaying, and storage tasks is commercially available. The CADDMAS is illustrated in Figure 1-1.

This paper deals with the design, implementation, testing and debugging of the Field Programmable Gate Array (FPGA) logic for a newly redesigned ADC mainboard and Peripheral Component Interconnect (PCI) interface board used for the ADC hardware. Each hardware components' functions had to be determined and the FPGA logic needed to perform these functions then had to be designed. Logic to increase the sample rate of the data acquisition from the analog-to-digital converters was designed. Logic was then developed to control a Universal Serial Bus (USB) 2.0 interface located on the ADC mainboard for data transmission into a computer. A packet based data transmission scheme and the supporting logic for the PCI interface board was also developed. All of the logic designs were simulated, debugged, and tested using a

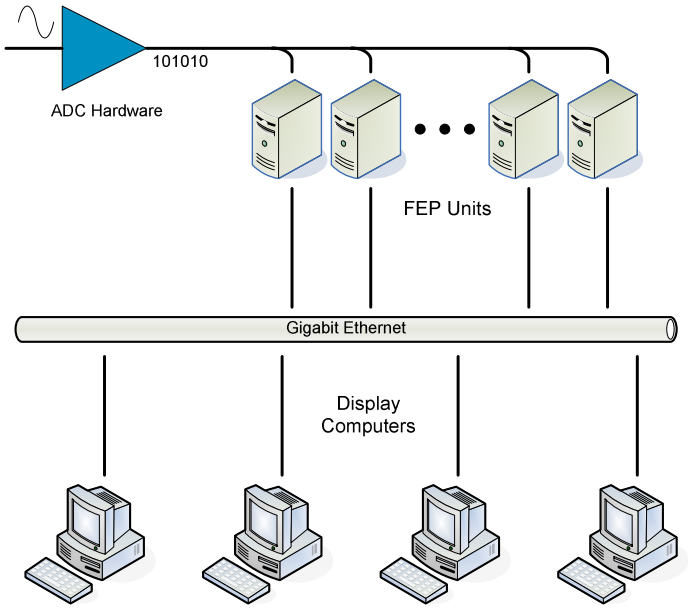

Figure 1-1: CADDMAS

commercial FPGA design software package. A software package was also developed to further test the FPGA logic designs and to demonstrate to a commercial software company how to incorporate the ADC hardware into their software. Using the software developed, a maximum data transfer rate of 96.15 Megabyte per second (MBps) was obtained from the PCI interface board as well as a transfer rate of 18.45 MBps from the USB interface. Chapter 2 will describe the original ADC hardware and its shortcomings, highlight the features present with the new ADC hardware, and the goals to be attained with this new hardware for this thesis. Chapter 3 will address the main topic of the FPGA designs for both the mainboard and the PCI interface board. Chapter 4 will describe the software written for the verification of the hardware and FPGA designs. Results from the verification software outlined in Chapter 4 will be presented in Chapter 5. Finally a summary of the work preformed and recommendations to improve upon the

design will be presented in Chapter 6.

CHAPTER 2
Background

2.1 ORIGINAL ADC HARDWARE

The original ADC hardware consisted of a mainboard, channel modules, and a PCI interface board. A block diagram showing its major components and connection is provided in Figure 2-1. Its hardware features and operation are described below.

The channel modules were removable from the mainboard and accepted a +/-5 volt differential analog input signal. They used a 64 times oversampling 16-bit sigma-delta Analog Devices AD7722 analog-to-digital converter capable of a maximum sample rate of 220,000 samples per second [1]. The module converted the analog signal into an asynchronous serial digital bit stream to be sent to the mainboard FPGA.

The mainboard allowed for 24 modules to be populated onto it and controlled the modules' sample rate. It was also responsible for reading the data from the modules and transmitting the data, via RS422, to the PCI interface board. It used an Altera FLEX 10K30 FPGA, which was the best FPGA available at the time of the original ADC hardware design, but contained no embedded random access memory (RAM) and relatively few logic resources by present standards [2]. This allowed for no buffering of the incoming module data and prompted the use of a less than optimal data transfer scheme. The transfer scheme simply consisted of a 512 bit shift register, running at 40 Megahertz (MHz), to shift the data in from the modules and then out to PCI interface board. Because 512 bits needed to be shifted out per sample and the shift register only ran at 40 MHz, this limited the overall sample rate of the system to 78,125 Hz. The use of this type of data transfer scheme was cumbersome when the mainboard was operated

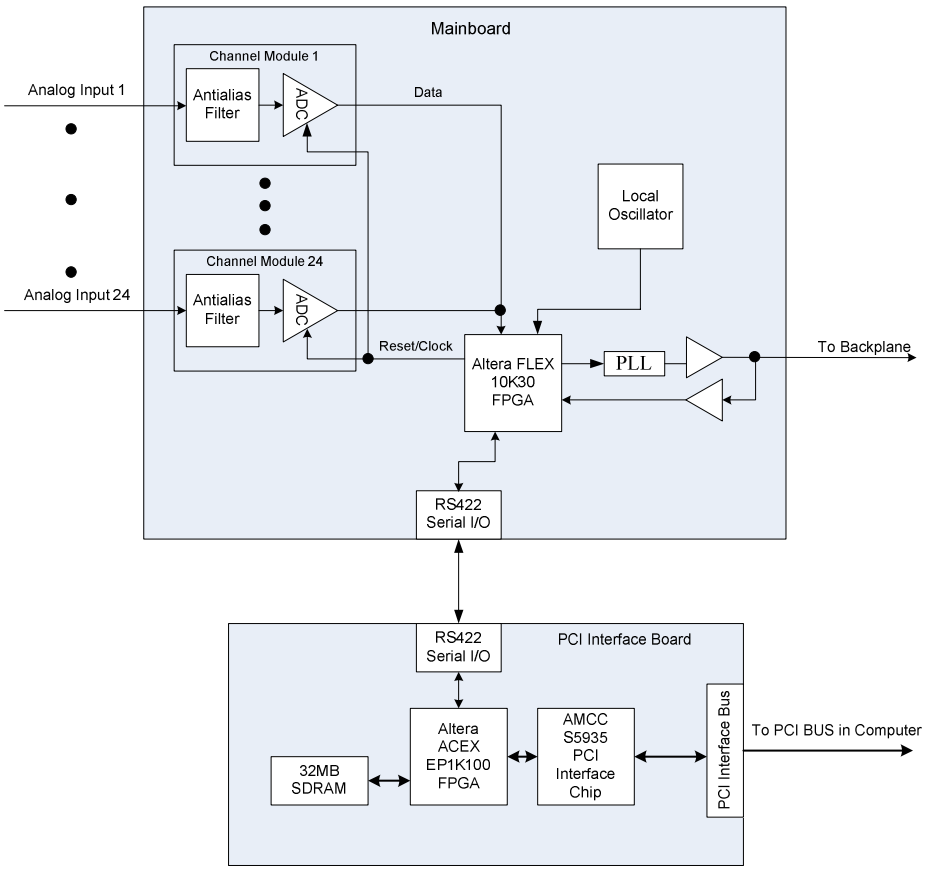

Figure 2-1: Original ADC Hardware

at sample rates slower than the maximum because redundant data transmissions would occur between sample times. For example, if the sample rate chosen for the system was 39,062.5 Hz, ½ of the maximum, only half of the 512 bits transferred to the PCI interface board would contain valid data samples. At ¼ the maximum sample rate, only one fourth of the 512 bits would be valid and so on. The result of this was additional software overhead on the PC to discard the invalid data and wasted direct memory access (DMA) time transferring invalid data.

The PCI interface board was responsible for initiating the data transfers and buffering the data into the computer. It also used a 512 bit shift register to read the data in from the mainboard but contained a 32 megabyte (MB) Synchronous Dynamic Random Access Memory (SDRAM) memory bank for buffering. It used an AMCC S5935 PCI interface chip to transfer data into the computer via its PCI bus [3]. This PCI interface chip lacked a standard software driver which made software development difficult. It was also a 5-volt logic device that was incompatible with newer PCs that only had 3.3-volt PCI slots.

This hardware possessed several shortcomings which prompted a redesign of the hardware. One of the major reasons to redesign the hardware was because FPGA logic resources and memory on the mainboard were either too limited or not present. This prohibited any significant changes to be made to the operation or transfer scheme of the mainboard. The limited performance and size of the FPGA on the mainboard also prohibited a target bandwidth of 50,000 Hz from being reached, as well as independent sample rates for banks of 8 channels each. Several components on the mainboard were also becoming obsolete and thus hard to obtain. A standard Windows® driver was also

not available for the PCI interface board making software development less than optimal.

The following lists some goals for this thesis project to overcome the aforementioned deficiencies. A maximum sample rate of 156,250 Hz on all channels of two mainboards should be obtained with reliable transmissions to the computer. A packet based data transfer protocol should be implemented to eliminate redundant data transmissions present with the original shift register approach. Independent sample rate control for each eight channel bank on the mainboard should be implemented to allow for maximum flexibility of data acquisition. A Universal Serial Bus (USB) 2.0 interface should be added as an alternate method of transferring data to the PC. This, in turn, requires a large memory buffer on the mainboard. A small verification and performance monitoring software suite to read the digital data from at least 2 mainboards into either the PCI interface board or USB 2.0 ports of the computer should also be developed. This minimal software package could then be given to a commercial software company for inclusion into their software so that the newly designed ADC hardware can be used by the CADDMAS at AEDC in the future. The next section will describe the newly designed ADC hardware and how its features allow for the goals just described to be accomplished.

2.2 NEW ADC HARDWARE

The previous section described some basic shortcomings of the previously used ADC hardware before it was redesigned. The following discussion will outline features present in the newly designed ADC hardware components for the CADDMAS to overcome those limitations. The new ADC hardware was designed by Dr. Bruce Bomar

of UTSI under contract in support of AEDC aeromechanical system upgrades which the work contained in this thesis also supported. The hardware is used to convert analog signals into digital data for further processing by the CADDMAS. The hardware consists of three components: channel modules, a mainboard, and PCI Interface card. A diagram illustrating the major hardware components and typical connections is shown in Figure 2-2. Although this thesis deals with the logic design and verification of the two FPGAs shown in Figure 2-2, it is necessary to understand the entire system to understand the FPGA design.

The only change that occurred to the channel modules was an improvement to the analog anti-alias filtering. The previous design simply used a single pole passive filter. This provided -13dB attenuation for out-of-band signals. This was updated to a 5 pole Bessel active filter to approximate linear phase response and to obtain -30dB out-of-band signal rejection. All other features such as the +/-5 volt differential analog input signal, the 64 times oversampling 16-bit sigma-delta ADC, and the asynchronous serial digital output bit stream remained unchanged.

The mainboard received the most attention during the redesign process since it is the heart of the data acquisition and transmission into the computer. It still contained sites for 24 modules but they were arranged as three banks of eight modules each. To accommodate the 156,250 Hz maximum sample rate a larger, more up-to-date Altera® Cyclone EP1C6 FPGA was incorporated onto the board [4]. This provided higher logic clocking frequencies as well as more logic resources and embedded RAM not available in the previous FPGA. The increased logic and memory resources allowed a packet based transmission scheme to be implemented to reduce redundant data transfers, as well

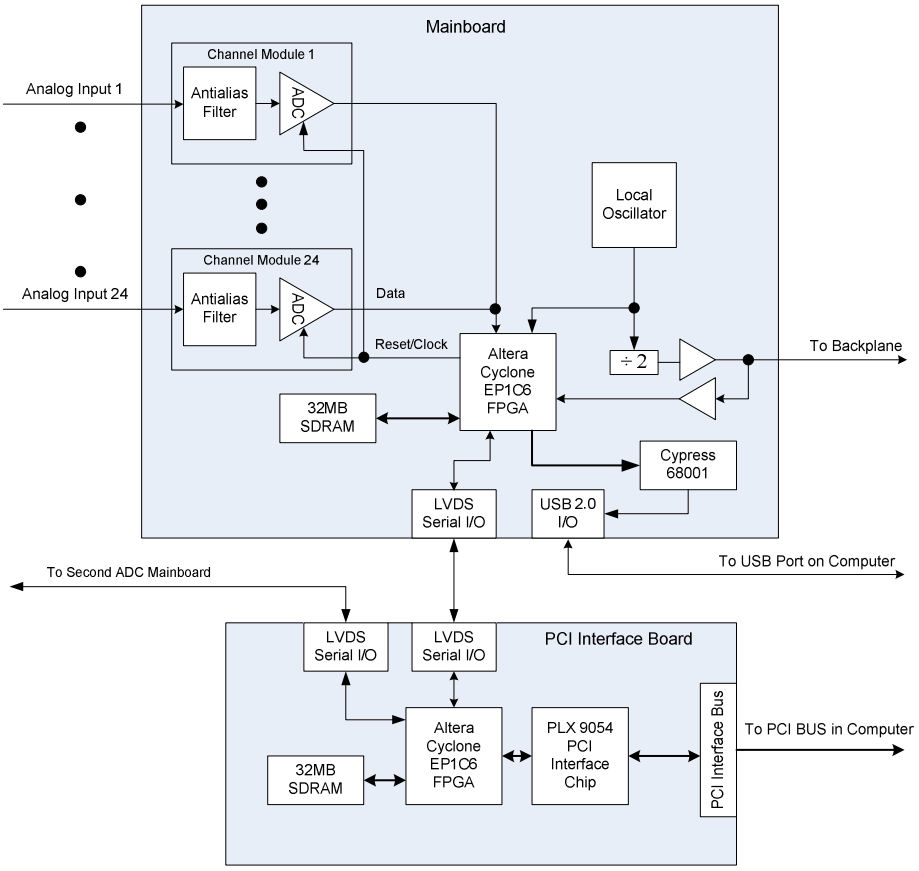

Figure 2-2: New ADC Hardware

as allow the implementation of independent clocking schemes for each of the eight channel banks present on the mainboard. For transmission into the computer, Low Voltage Differential Signaling (LVDS) drivers, which are capable of 400 Mbps, were chosen instead of the previous RS422 drivers, which are limited to 20 Mbps. The increase of data transmission capability was necessary to allow for the increase in sample rates and the packet based protocol. A USB 2.0 interface, capable of 480 Mbps, was also added to allow for easier connection and transmission into the computer for smaller, more compact systems. However, even though USB 2.0 and LVDS have similar data transfer rates, the LVDS to PCI interface option was retained since it allows much more distance between the PC and mainboard than is possible with USB. The USB 2.0 interface was provided with the Cypress Semiconductor 68001 USB 2.0 chip [5]. It included a Windows® driver and an Application Programming Interface (API), CyAPI, [6] for easy inclusion into a software program. For onboard buffering of the USB 2.0 data, a 32 MB SDRAM chip was chosen. To further enhance data quality, a low jitter 20 MHz oscillator used to clock the modules and FPGA logic was also incorporated onto the mainboard. A hardware jumper was used to designate if the mainboard is a system controller. If the mainboard is a controller, its oscillator output, divided by 2, as well as a reset signal from the FPGA is fed onto a backplane. This allows multiple synchronized mainboards to be incorporated into a single system and have all channels in the system begin sampling at the same time from one master clock. This is important if time correlation among channels and relative phase information is needed.

Finally a substantial redesign to the PCI interface board was conducted. To acquire a standard Windows software driver, a commercially available PCI bus mastering

interface chip was used on the new design. The PLX® PCI9054 provided a 32-bit, 33 MHz PCI bus into the computer, as well as a Windows driver and API, PlxApi [7], to be used in a Windows software program. A FPGA in the same Altera® EP1C6 family was chosen for its high logic density and high clocking rates. A 32 MB SDRAM chip was used for onboard buffering of the serial LVDS data coming from the mainboards before being transmitted into the computer. An LVDS receiver was also used to maintain the electrical specification of the transmission medium into the computer and to accommodate the high data rates.

All of these improvements to the hardware allowed for considerable flexibility in the system configuration and increased data throughput into the computer. The rest of this document will concentrate on the FPGA logic designs and verifications on both the ADC mainboard and PCI interface board.

CHAPTER 3
FPGA DESIGNS

3.1 DEVELOPMENT SOFTWARE

Since both the mainboard and PCI interface board used an Altera® FPGA, an Altera software program, called Quartus® II, was chosen for the development of the FPGA logic designs [8]. Quartus II provides an entire development environment, for the designing, simulating, compiling, and programming of their FPGAs. Another reason to use an Altera based software package for development is that the software has many built-in logic functions such as adders, multiplexers, shift registers, etc. These are referred to as megafunctions within Quartus II. These reduce development and debugging time for the design by using available pre-tested components and were used extensively throughout the FPGA designs. Any other logic such as state machines needed by the FPGA designs were created using Very High Speed Integrated Circuit Hardware Description Language, VHDL, and the graphical interconnect tools provided by the Quartus II software.

3.2 ADC MAINBOARD LOGIC DESIGN

This section will detail the FPGA design process and operation for the mainboard. A block diagram illustrating the various subsystems for the FPGA design is shown in Figure 3-1. The roles of each subsystem within the total design are described throughout this subsection.

A block diagram of the clock generation subsystem is provided in Figure 3-2. This subsystem provides the sampling clocks for the modules as well as all of the utility

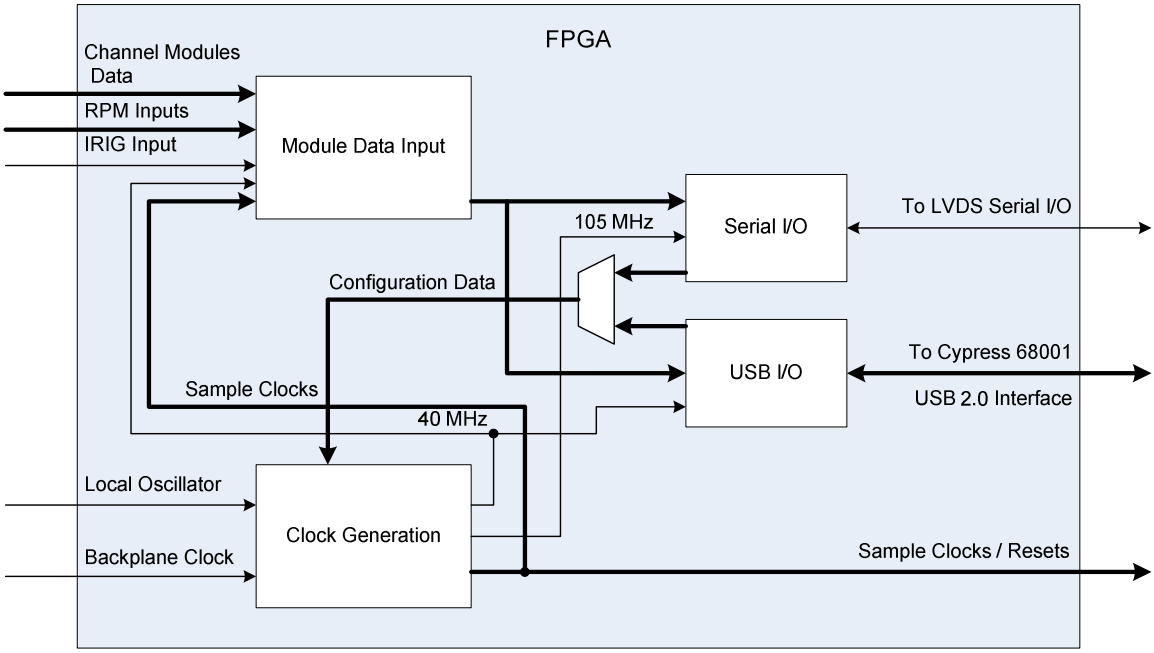

Figure 3-1: Overall ADC FPGA Block Diagram

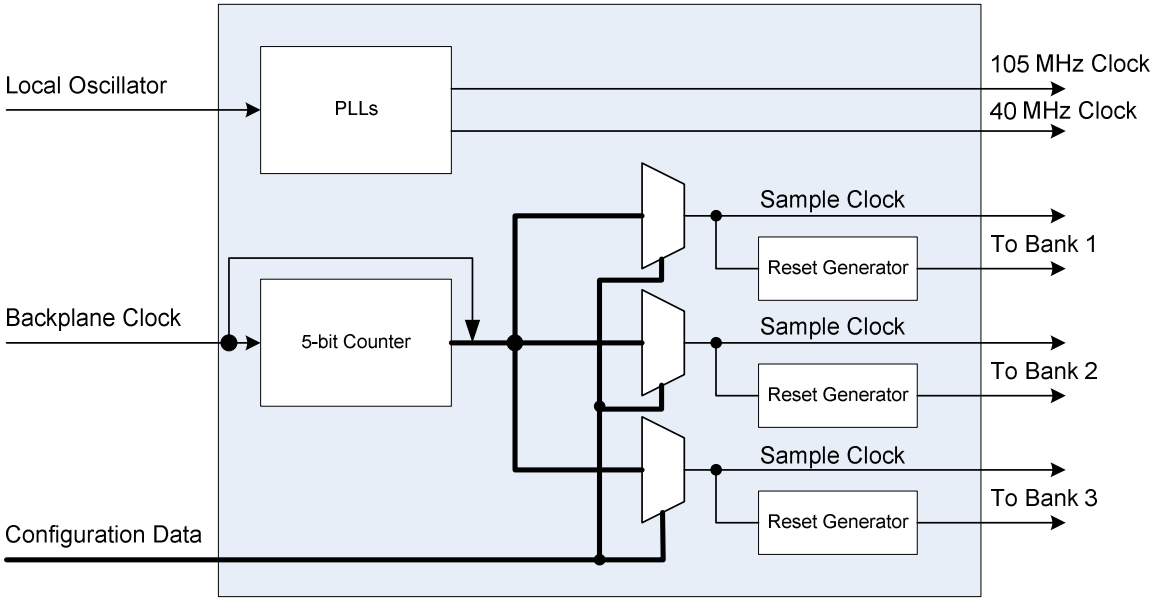

Figure 3-2: ADC Clock Generator Block Diagram

clocks used throughout the FPGA. The utility clocks are generated from the onboard oscillator by the use of phase locked loops (PLL) located within the FPGA. In order to program the output frequencies from the PLLs, a PLL megafunction was used for each PLL. 105 MHz and 40 MHz clocks are generated by the PLLs and distributed to the various subsystems in the FPGA. The sampling clocks for each of the three banks are generated from the clock received from the backplane connection. By using a 5-bit counter, the backplane clock was divided down to generate 6 different clocking frequencies which are synchronous binary multiples of each other. For each eight channel bank, a 6-to1 multiplexer was used to select the desired clocking frequency. Because the modules use a 64 times oversampling sigma-delta ADC, a clock 64 times higher in frequency than the desired sample rate must be sent to each of the modules. The required clocking frequencies and resulting sampling rates are outlined in Table 3-1. This subsystem also generates a reset pulse for each individual ADC bank to allow all channels in a bank to be synchronously sampled.

The module data input subsystem contains all of the logic necessary to read the data into the FPGA from the channel modules and store the received data into internal memory buffers. It consists of three module receiver units, three 29-bit engine

Table 3-1: Clocking Frequencies

Clocking Frequency (Hz)	Sample Rate (Hz)
10,000,000	156,250
5,000,000	78,125
2,500,000	39,062.5
1,250,000	19,531.25
625,000	9,765.625
312,500	4,882.8125

revolutions per minute (RPM) frequency counters to meet the requirements of multiple compressor / turbine stages of an aircraft engine, three 29-bit packet counters, an Inter-Range Instrumentation Group (IRIG) time decoder, some multiplexers and a controlling state machine. Its basic block diagram is shown in Figure 3-3.

The module data input subsystem is responsible for reading data from the various data sources (module receivers, RPM counters, etc.) and assembling the data into packets. The packets are then buffered into FIFOs located in the serial I/O and USB I/O subsystems to allow for transmission out of the mainboard and into the computer. The packet structure is shown in Figure 3-4. A packet contains the data from only one eight channel bank of modules, plus the IRIG time stamp and either an RPM frequency count or packet number. Every packet is time stamped but because of the high sample rates of the system, RPM counts (which are of very low frequency) only need to be sent out occasionally. Packet numbers are added to the packet when an RPM count is not going to be sent in order to detect lost packets during transmission to the computer. For example, consider an engine running at 12,000 RPM. Assuming a RPM based on an once-per-revolution pulse, this results in a new RPM count being generated 200 times a second. Even at the lowest sample rate of 4,883 Hz an RPM count is only added to approximately every 24^{th} packet. The rest of the packets will contain an incrementing packet number to allow for quick detection of failed data delivery into the computer. A 2-bit bank index is used to indicate which of the three banks this packet of data is from. 2 bits from a 3-bit type field are also included in the packet to indicate if the 29-bit count is the first, second, or third RPM value or simply a packet number. The other bit in the type field is used to indicate if the packet is from the first or second ADC mainboard fed into

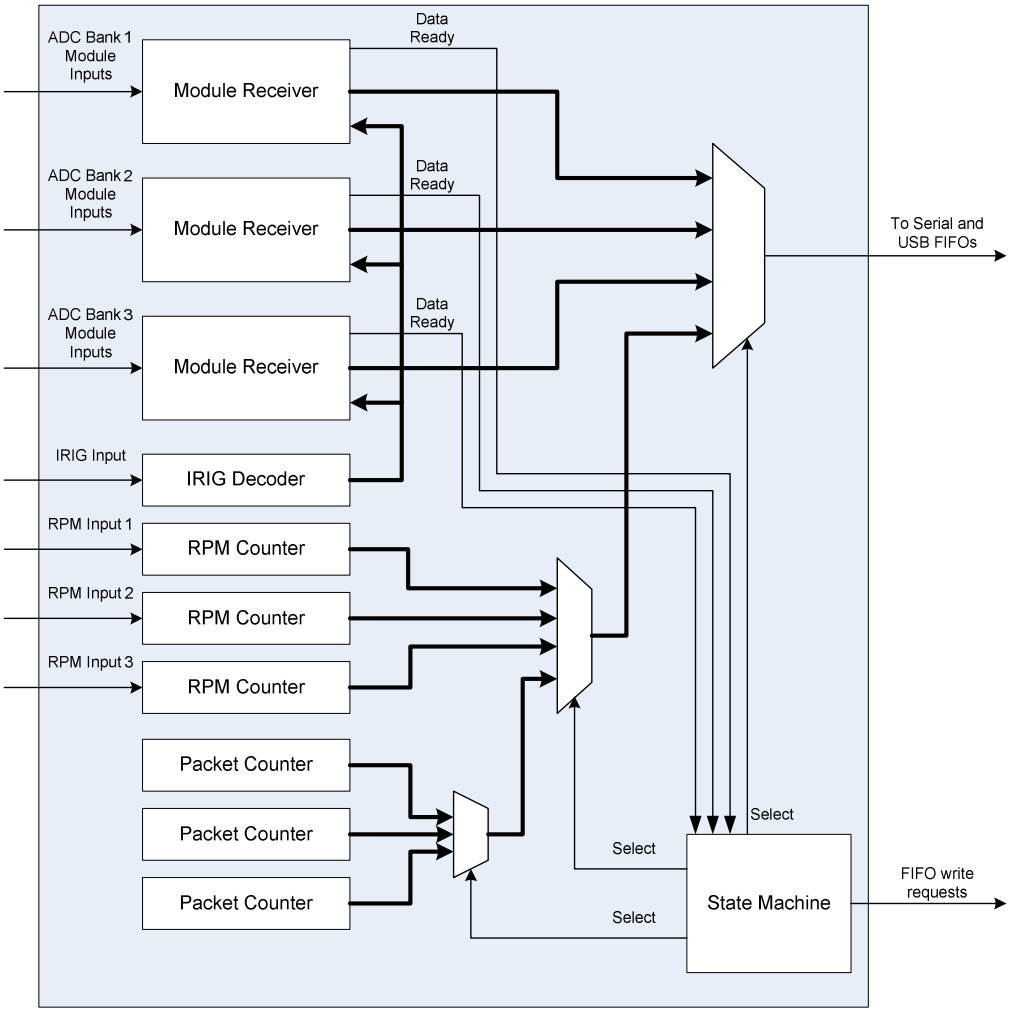

Figure 3-3: ADC Module Data Input Block Diagram

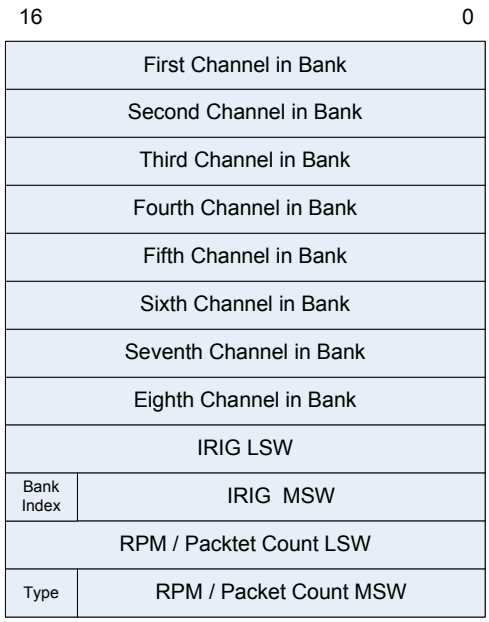

Figure 3-4: ADC Data Packet Structure

the PCI interface board. This bit is set in the PCI interface board when a packet of data is read into the PCI interface board. Each packet contains 24 bytes of data. When all three banks are running at the maximum sampling rate of 156,250 Hz, this correlates to a transfer rate of 3*156,250*24 = 11.25 Megabytes per second (MBps). The 29-bit RPM counters operate at a clock frequency of 40 MHz and function as free running counters. When an RPM pulse is received, the current counter value is latched into a 29-bit D flip-flop register and a flag is set to notify the module data input state machine that an updated RPM count is available. The IRIG decoder, which also operates at a clocking frequency of 40 MHz, was provided from a previously used design [9] and outputs 30 bits of IRIG time information. The 29-bit packet counters operate at 40 MHz and are incremented by the controlling state machine once the current packet count is transmitted into the FIFOs.

When the module receivers finish collecting a sample from the eight channels in a bank they notify the module data input state machine that they have data ready to be sent to the computer. The module data input state machine, which operates at 40 MHz, selects the receiver and begins to transfer the channel data and IRIG time stamp on a byte per byte basis from the FIFOs located in the module receiver into the FIFOs contained in both the serial input/output and the USB subsystems. Upon completing the data transfers, the state machine then determines if any RPM count is ready to be sent to the computer or simply a packet number. It selects the appropriate type of count value and proceeds to store that information on a byte-per-byte basis in the FIFOs as well. If an RPM count was sent, the flag set by the RPM counter to indicate that a new RPM count was available is cleared by the controlling state machine. If a packet count was sent, the appropriate packet counter is incremented by the state machine. That completes the buffering of one packet of bank data. The state machine then waits for another notification from a module receiver and the process is repeated.

The three module receiver units provide the interface to the digital bit stream produced from the channel modules. This unit uses the sample clock for the bank it is connected to as its clock source because all of its operations are synchronous to the channel modules. Each module receiver consists of eight 16-bit shift registers, a 20x8 bit multiplexer, a 512x8 bit FIFO, and a controlling state machine. A diagram illustrating this unit is shown in Figure 3-5. The channel modules' digital bit stream is constructed as an asynchronous serial link with a start bit, 16 data bits, and 15 stop bits transmitted at half of the sample clock rate. The controlling state machine examines the data from the first channel in the first bank for a start bit. When a start bit is detected, the state machine

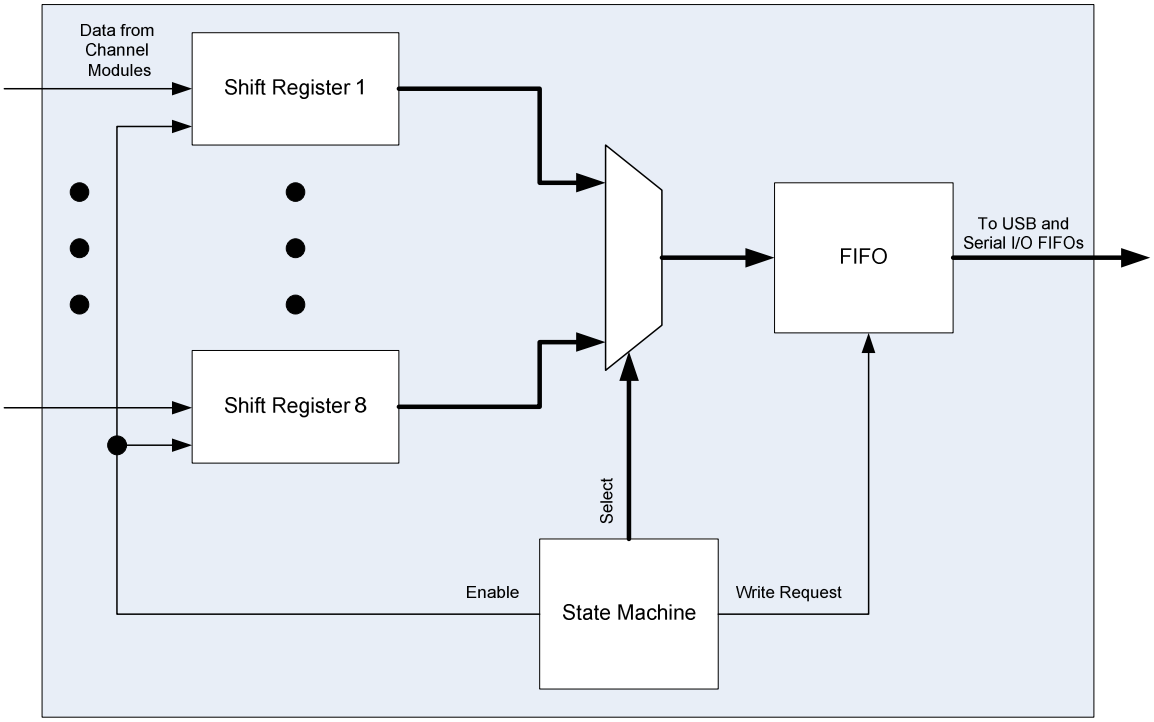

Figure 3-5: ADC Module Receiver Block Diagram

begins to enable the shift registers every other clock cycle for 32 cycles to store the sampled data. After this has occurred the state machine has 30 more clock cycles to store all of the just received data into a FIFO. It selects each channel shift register through the 20x8 bit multiplexer and initiates write requests to the FIFO on a byte per byte basis. Upon completely writing all of the data from the shift registers, it then writes the 4 bytes of data from the IRIG decoder into the FIFO as well. Finally it waits again for another start bit and the process repeats.

The serial I/O subsystem, operating at 105 MHz, consists of a 2048x8 bit FIFO, controlling state machine and an 8-bit output shift register. Its block diagram is shown in Figure 3-6. The controlling state machine monitors the FIFO to determine if the FIFO is empty. If the FIFO is not empty, a byte of data is ready to be transferred to the PCI

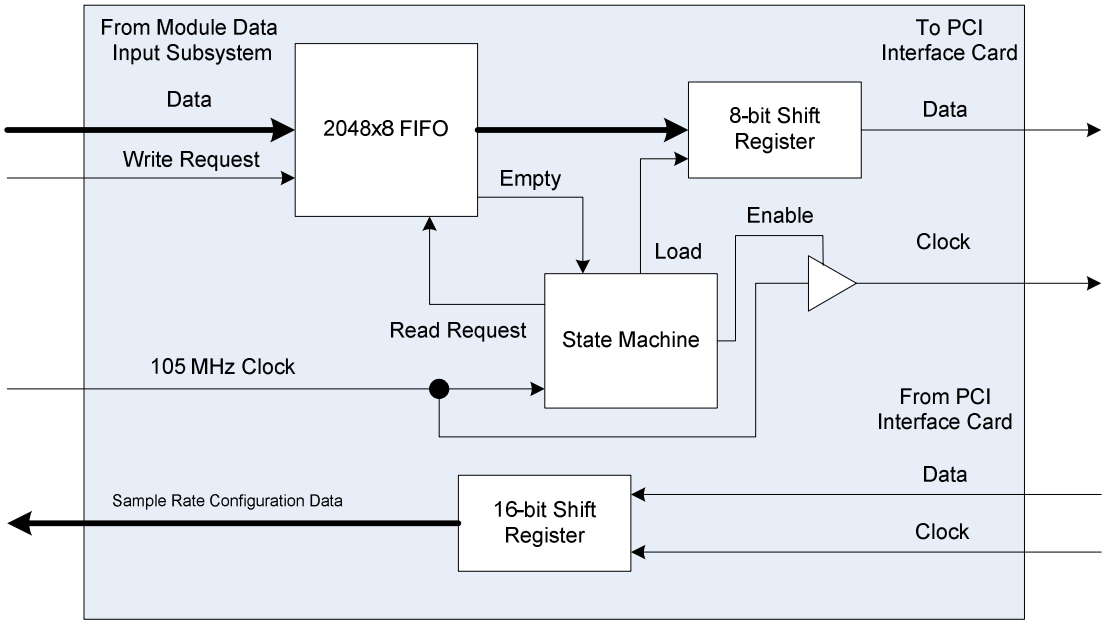

Figure 3-6: ADC Serial I/O Block Diagram

interface card. The state machine proceeds to read the byte of data out of the FIFO and load it into the shift register. Each byte transmitted requires 9 clock cycles to complete the transmission. The first clock cycle is used to load the shift register and read the next byte value out of the FIFO. The following 8 clock cycles are used to shift the byte out of the shift register. It writes the bytes of data into the output shift register and enables it so that the data is shifted out of the mainboard, at a rate of 105 MHz, and into the PCI interface board. The 105 MHz shift register clock is also transmitted to the PCI interface board so that it can receive the serial data transmitted to it. After a packet's worth of data is sent out of the mainboard, the state machine then begins to examine the FIFO again and the process repeats. The serial I/O subsystem also contains an input shift register which is used to configure the mainboard after power on reset. At power on reset, the default interface to the computer is set to the serial I/O subsystem. The PCI interface

board transmits 16 bits of sample rate configuration information to the mainboard so that the mainboard knows at which sample rates to run the banks. The configuration information is illustrated in Figure 3-7.

Each of the three 8 channel banks use 3 bits to determine which of the 6 supported sample rates it will run at and 1 more bit to determine if it is enabled or not. Table 3-2 documents the meanings of each of the 16 bits contained in the configuration input word.

The USB I/O subsystem provides all of the controls for the USB 2.0 and SDRAM interfaces. All data interfaces are 8 bits wide. It operates at 40 MHz and contains a SDRAM controller, 2048x8 bit SDRAM input FIFO, a 1024x8 bit SDRAM output FIFO, a 31x8 bit USB configuration ROM, a 2x8 bit multiplexer, a 16-bit D flip-flop sample rate configuration register, two bi-directional 8-bit buffers and a controlling state machine. A block diagram for it is shown in Figure 3-8. Upon power-on reset, the controlling state machine accesses the configuration ROM and selects the bi-directional buffer to write data from the configuration ROM to the Cypress Semiconductor USB 2.0 interface chip. It proceeds to write the configuration data to the Cypress Semiconductor chip necessary to prepare the USB interface for data transfers. The Cypress semiconductor 68001 contains internal 4 kilobyte FIFOs for both its input and output interfaces with full and empty flags for both. Upon completion of downloading the

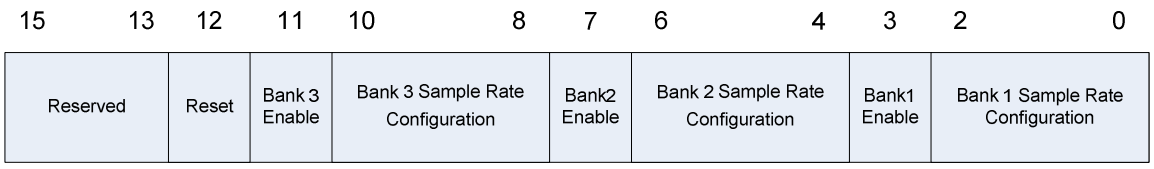

Figure 3-7: ADC Sample Rate Configuration Data

Table 3-2: Configuration Data Description

Reserved	Reset	Bank 3 Enable	Bank 3 Sample Rate Configuration
Unused	0 = Board Reset	0 = Disabled	000 = 4,882.8125 Hz
	1 = Board not Reset	1 = Enabled	001 = 9765.625 Hz
			010 = 19,531.25 Hz
			011 = 39062.5 Hz
			100 = 78,125 Hz
			101 = 156,250 Hz

Bank 2 Enable	Bank 2 Sample Rate Configuration	Bank 1 Enable	Bank 1 Sample Rate Configuration
0 = Disabled	000 = 4,882.8125 Hz	0 = Disabled	000 = 4,882.8125 Hz
1 = Enabled	001 = 9765.625 Hz	1 = Enabled	001 = 9765.625 Hz
	010 = 19,531.25 Hz		010 = 19,531.25 Hz
	011 = 39062.5 Hz		011 = 39062.5 Hz
	100 = 78,125 Hz		100 = 78,125 Hz
	101 = 156,250 Hz		101 = 156,250 Hz

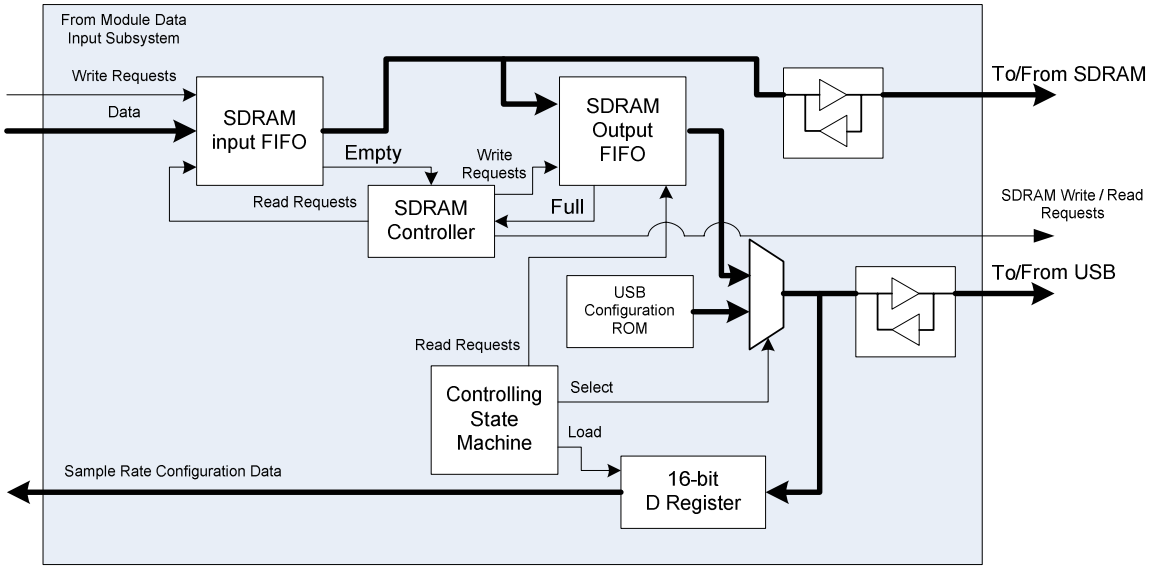

Figure 3-8: ADC USB I/O Block Diagram

configuration information to the 68001, the state machine monitors the empty flag on the USB internal input FIFO to see if a PC has placed data into the FIFO. Upon power-on reset, the serial I/O subsystem is the default interface connected to the computer. It is necessary for the USB I/O subsystem to receive its sample rate configuration data via the USB. When the empty flag indicates that data is present in the input FIFO, the state machine stores the sample rate configuration data transmitted from the computer into the 16-bit D flip-flop register. The sample rate configuration data is identical to the data sent from the PCI interface board to the serial I/O subsystem. After 16 bits of data are read from the computer, the USB I/O subsystem selects its sample rate configuration to be used by the mainboard. This information is then used to program each of the three banks.

The SDRAM controller logic was provided by a previous design [10] and provides an 8-bit data interface to both the input and output FIFOs as well as to the SDRAM itself. The SDRAM provides 32MB of memory buffering. The SDRAM controller monitors an empty flag from an input FIFO, which the module data input subsystem writes to, and reads the data from the FIFO when it is not empty. It then writes the data into SDRAM. An output FIFO is also monitored by the SDRAM controller for being full, and if it is not full data is written into the output FIFO from the SDRAM. The SDRAM output FIFO provides an empty flag to the USB state machine. After the sample rate configuration data is sent from the computer, the state machine then monitors the full flag on the internal USB output FIFO. If the USB output FIFO is not full and the SDRAM output FIFO is not empty, the state machine reads the data from the SDRAM output FIFO and writes the data into the internal USB output FIFO. The USB state machine then alternates between checking for an empty flag from the USB internal

FIFO and a full flag from the USB internal output FIFO. If the internal USB FIFO is not empty, the USB state machine reads new sample rate configuration data from the computer. Otherwise the state machine checks if the internal USB output FIFO is not full. If the USB output FIFO is not full more data is written to the USB FIFO and the process of checking both the USB input and output FIFOs is repeated.

This section outlined the FPGA design used on the ADC mainboard. All of the subsystem designs were detailed to illustrate how each perform a specific function to perform the overall operation of getting the data from the ADC modules into the computer either through the PCI interface card or the USB interface. The next subsection will describe the design used for the PCI interface card necessary to read data into the computer via the PCI bus.

3.3 PCI INTERFACE LOGIC DESIGN

This section will describe the FPGA design employed to program the PCI interface card to read data from the ADC mainboard into the computer via the PCI bus. The FPGA reads data from the LVDS receivers and transmits it to the PLX 9054 PCI interface chip. The PLX chip then transmits the data into the computer via the PCI bus. The FPGA also reads sample rate configuration data sent from the computer via the PCI bus and PLX interface chip and transmits it to the ADC mainboard. The PCI interface board FPGA design consists of a serial input subsystem, a serial output subsystem, a 2048x16 bit SDRAM input FIFO, a SDRAM controller, a 2048x16 bit SDRAM output FIFO, two bi-directional buffers, and a controlling state machine. A block diagram

showing the subsystems contained within the FPGA is given in Figure 3-9. Each subsystem's design and operation is presented below.

The clocks for each of the subsystems are generated using the built-in Altera PLL megafunction. The PLL is programmed to generate both a 50 MHz and a 70 MHz clock which is then distributed throughout the FPGA design. The 50 MHz clock is also used to synchronize the FPGA to the PLX PCI9054 chip.

The serial input subsystem is responsible for reading data in from the ADC mainboard. It consists of a two 8-bit shift registers, two 8-bit D flip-flop registers, two 2048x16-bit FIFOs, a 2x16-bit multiplexer and three controlling state machines. A block diagram is shown in Figure 3-10. Each ADC mainboard operates independently of others except for a common sample clock and reset signal. After reset, the serial data is shifted into the corresponding 8-bit shift register using the serial clock while the corresponding controlling state machine monitors how many bits are shifted in. After 8 bits are shifted in, the controlling state machine enables the D flip-flop register for 1 clock cycle to latch the data just shifted in. After the next 8 bits are shifted into the shift register, the state machine issues a write request to the corresponding FIFO and all 16 bits are written into the FIFO. The process is then repeated indefinitely until another system reset is issued.

While data is written into both of the FIFOs, the main serial state machine monitors the contents of both of the FIFOs. When the main serial state machine detects that a FIFO has an entire 24 byte data packet, it selects the corresponding FIFO through the multiplexer and issues 12 write requests to the SDRAM input FIFO in order to write all 24 bytes, 16 bits at a time, into the SDRAM input FIFO. If the packet of data is read from the FIFO connected to the second ADC mainboard, the main serial state machine

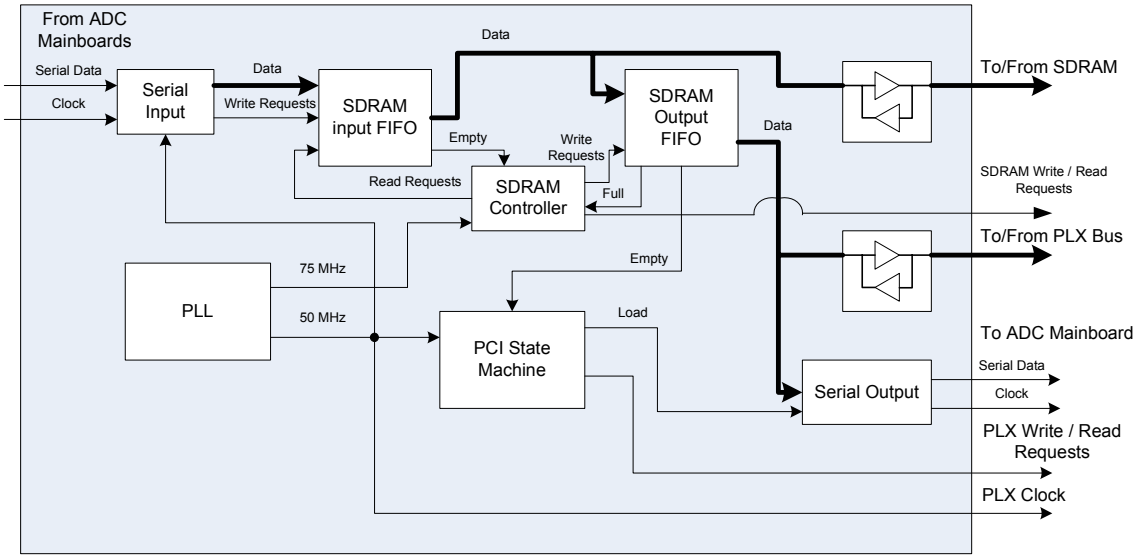

Figure 3-9: Overall PCI Interface Board FPGA Block Diagram

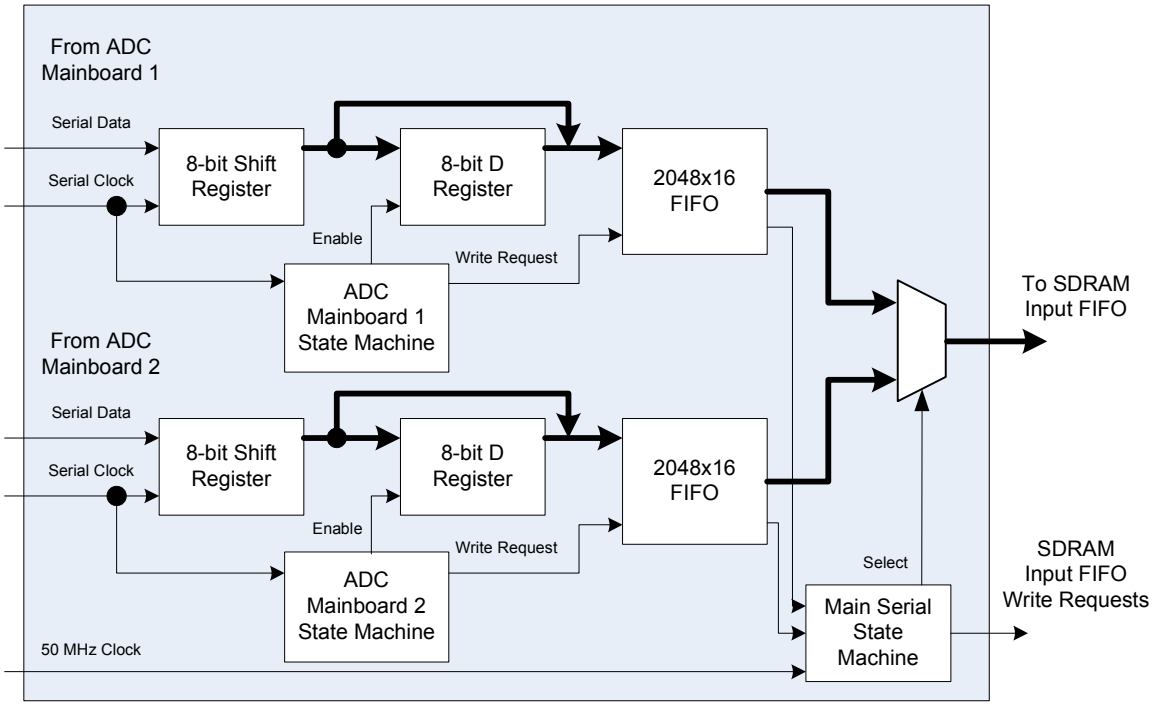

Figure 3-10: PCI Interface Board Serial Input Block Diagram

sets the third bit in the type field of the data packet to indicate to the computer software that the data packet was sent from the second ADC mainboard. The main serial state machine alternates monitoring each of the FIFOs and when a FIFO has a 24 byte packet, it issues write requests to the SDRAM input FIFO for further storage into the SDRAM buffer.

The SDRAM interface logic was provided from a previous design [10] and is identical to the one used in the ADC mainboard except that is uses a 16-bit data interface to both the input and output FIFOs as well as to the SDRAM instead of an 8-bit interface used on the ADC mainboard. It also operates at 75 MHz to handle the data rates for two ADC mainboards running at the highest sample rate. The SDRAM input FIFO provides a flag to notify the SDRAM controller that data is ready to be written into the SDRAM. The controller monitors this flag and when it is set it writes the data into the SDRAM through the bi-directional buffer. The SDRAM output FIFO is also monitored by the controller for being full. If the output FIFO is not full, data is read from the SDRAM through the bi-directional buffer and written into the output FIFO. The output FIFO also provides an empty flag to the PCI state machine to indicate data is ready to be sent to the PCI interface chip.

The PCI state machine, which operates at 50 MHz, interfaces to the PLX PCI9054 chip, the SDRAM output FIFO and the serial output module. All data transfers to and from the PCI bus are initiated by the PLX chip. After reset, the PCI state machine waits for a read data request from the PCI interface chip. This is the sample rate configuration data. Since up to two ADC mainboards can be connected to the PCI interface board, 32-bits are needed to configure both of them. The state machine reads the 32 bits of data

from PCI9054 through a bi-directional buffer and loads the data into a register located in the serial output subsystem. The serial output operation is discussed later. After the sample rate configuration data is received from the computer, the PCI state machine then waits for a data write request from the PCI9054. Once a write request is received, it checks the empty flag provided from the SDRAM output FIFO to determine that data has been sent from the ADC mainboards. The state machine waits until the SDRAM output FIFO is not empty and then reads data from the SDRAM output FIFO. It writes the data, via the bi-directional buffer, to the PLX PCI9054 chip so that it can be transferred into the computer via the PCI bus. It then waits for either a read or write request from the PCI9054. It performs the appropriate action of either reading data from the PLX chip and storing it into the serial output subsystem or writing data to the PLX chip from the SDRAM output FIFO.

The serial output subsystem provides all of the circuitry to interface to two ADC mainboards' serial inputs. It operates at 50 MHz and consists of a 32-bit D flip-flop register, two 16-bit shift registers, and a controlling state machine. Its block diagram is shown in Figure 3-11. When a read request from the PLX chip is processed by the PCI state machine, 32 bits of sample rate configuration data is loaded into the D flip-flop register by the PCI state machine. The load pulse is used to notify the serial output state machine that there is data to be sent to the ADC mainboards. The serial output state machine then loads each of the 16-bit shift registers, enables the serial clock output and proceeds to clock the data out to the ADC mainboards. Once all 16 bits have been shifted out, the serial clock is disabled and the state machine then waits for another load pulse from the PCI state machine.

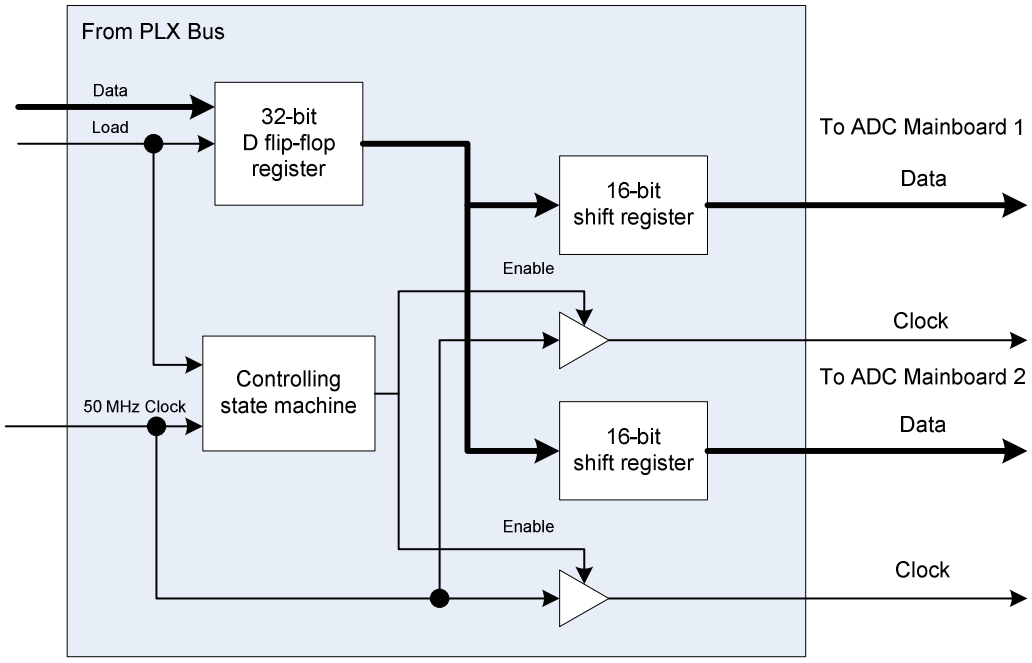

Figure 3-11: PCI Interface Board Serial Output Block Diagram

This concludes the description of the FPGA logic designs of both the ADC mainboard and the PCI interface board. Each FPGA has been carefully designed with simple components to provide a straightforward operation as well as to provide optimum performance. The next chapter will outline a small software package that can communicate to both the PCI interface board as well as to the USB interface provided by the ADC mainboard. It will be used to determine correct operation of both FPGA designs in addition to testing the maximum performance able to be obtained from both data transfer interfaces.

CHAPTER 4
VERIFICATION SOFTWARE

4.1 DEVELOPMENT TOOLS USED

All of the verification software was written in the ANSI C language. The USB 2.0 interface used by the ADC mainboard and the PCI interface board both provided Windows XP software drivers in addition to APIs for inclusion into a software program. Both of the software drivers were loaded onto the computer running the verification software to allow communication to both the PCI interface board and USB 2.0 interface located on the ADC mainboards. The hardware connections used to verify the FPGA designs in both the ADC mainboard and PCI interface board is shown in Figure 4-1.

The API provided by the PLX PCI9054 chip is called PlxApi [7]. It provides C language functions to communicate to the PLX PCI9054 chip and perform various operations on the PCI9054 such as opening the PCI interface board for reading and

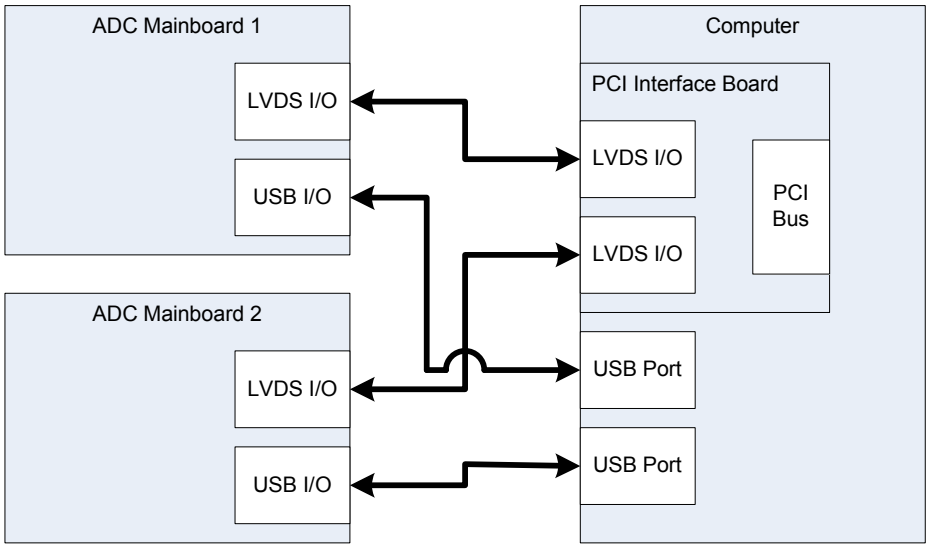

Figure 4-1: Verification Software Hardware Connections

writing, resetting the PCI interface board, initiating data transfers, and closing the PCI interface board. A summary of the PlxApi functions as well as a description of their purpose used in the verification software is outlined in Table 4-1.

The API provided by the Cypress Semiconductor 68001 chip is called CyApi [6]. It provides C language functions to open the Cypress Semiconductor 68001 chip for reading and writing as well as closing the devices after all data transfers are finished. A summary of the CyApi functions as well as a description of their purpose used in the verification software is outlined in Table 4-2.

Table 4-1: PlxApi Functions

Function Name	Purpose in Program
PlxPciDeviceOpen	Opens PCI interface board for data communication
PlxPciDeviceClose	Closes PCI Interface board
PlxPciBoardReset	Sends a hardware reset to the PCI interface board via the PCI bus
PlxIntrEnable	Enables various interrupts for the PLX chip
PlxIntrAttach	Attaches the interrupt to a software event
PlxIntrWait	Waits for an interrupt to occur
PlxIntrDisable	Disables various interrupts for the PLX chip
PlxPhysicalMemoryAllocate	Allocates a memory buffer to be used byt the PLX chip to store data
PlxPhysicalMemoryMap	Maps the memory buffer to a PCI address space
PlxPhysicalMemoryFree	Deallocates a memory buffer previously created by PlxPhysicalMemoryAllocate
PlxPhysicalMemoryUnmap	Unmaps a memory buffer from the PCI address space
PlxDmaBlockChannelOpen	Opens a DMA channel for block transfers
PlxDmaBlockChannelTransfer	Performs a DMA block data transfer to or from the PCI bus
PlxDmaBlockChannelClose	Closes a DMA channel for data transfers

Table 4-2: CyApi Functions

Function Name	Purpose in Program
Open	Opens the USB device for data transfers
DeviceCount	Determines the number of USB devices attached to the computer
XferData	Transfers data to/from computer
Close	Closes the USB device

4.2 VERIFICATION SOFTWARE OPERATION

Using the software drivers provided for both data interfaces and the API functions outlined above, the verification software allows for data to be read into the computer from the ADC mainboards for further storage and analysis. A software flowchart is shown in Figure 4-2. Upon start of the software, a menu is provided to the user to choose whether the PCI interface board or the USB interface on the ADC mainboards will be used for data communications. Once the data interface is chosen, the software essentially performs the same operation just using different functions from the PLX or Cypress APIs.

If the PCI interface board is chosen for the data interface, the PCI interface board is opened for data transfers using the PlxPciDeviceOpen function. Memory buffers that contained an integral number of ADC mainboard data packets are allocated using the PlxPhysicalMemoryAllocate and PlxPhysicalMemoryMap functions next. A DMA completion interrupt is then enabled using the PlxIntrEnable and PlxIntrAttach functions. The PlxPciBoardReset function is used to issue a PCI bus hardware reset to the PCI interface board. Once this is accomplished, a 32-bit sample rate configuration data word consisting of only zeroes to reset both of the ADC mainboards is written to the PCI interface using the PlxDmaBlockChannelOpen and PlxDmaBlockChannelTransfer functions. This completely resets both ADC mainboards connected to the PCI interface board. A 32-bit sample rate configuration word containing the desired sample rate data for both ADC mainboards is then written to the PCI interface board using the same PlxDmaBlockChannelOpen and PlxDmaBlockChannelTransfer functions. This allows the ADC mainboards to begin sending data to the PCI interface board at the selected

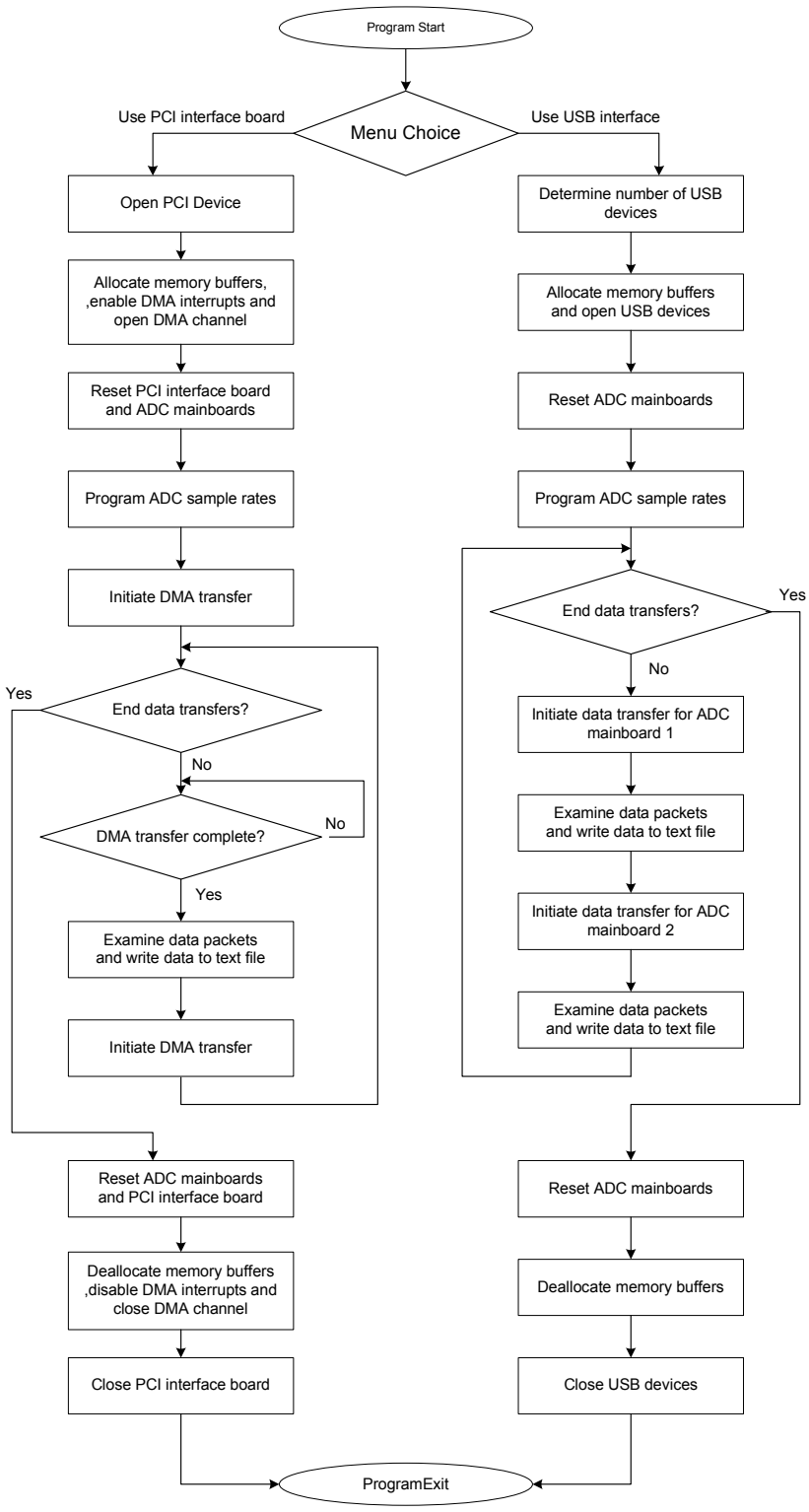

Figure 4-2: Verification Software Flowchart

sample rates. The software then calls the PlxDmaBlockTransfer function to begin reading data into the memory buffers. The software then enters a data transfer loop which is terminated when the user presses any key. The PlxIntrWait function is used inside the loop to wait for the DMA transfer to complete. Once the data transfer has completed, the memory buffer is transformed into 24 byte data packets which are examined for data integrity. If data packet corruption is detected by examining the packet numbers, the loop is automatically terminated. Otherwise data packets are then written to an output text file. After this is complete, another DMA transfer is initiated and the loop is repeated. To measure the data transfer rate obtained from the ADC mainboards, each data transfer is timed. The time it takes for a transfer to complete is then divided by the total bytes transferred to calculate the data transfer rate. The data transfer loop is terminated if the user presses a key. Once the data transfers are terminated, the ADC mainboards are reset by writing a zeroed 32-bit sample rate configuration word to them. This stops all data transfers from the ADC mainboards. The PCI interface board is then reset using the PlxPciBoardReset function. The memory buffers are freed from computer memory using the PlxPhysicalMemoryFree and PlxPhysicalMemoryUnmap functions. Following the memory deallocation, the DMA completion interrupt is disabled using PlxIntrDisable and the DMA channel is closed for data transfers using the PlxDmaBlockChannelClose function. Finally the PCI interface board is closed using the PlxPciDeviceClose function and the program exits.

If the USB interface is chosen by the user for the data interface, the CyApi DeviceCount function is used to determine the number of ADC mainboards connected to the computer. Memory buffers containing an integral number of ADC mainboard data

packets using the standard malloc() C function are allocated for the data transfers from both of the ADC mainboards. Each USB device is then opened using the Open function. A 16-bit sample rate configuration word consisting of complete zeroes is then written, using the XferData function, to both ADC mainboards to reset them. Once both of the ADC mainboards attached to the computer have been reset, a 16-bit sample rate configuration word representing the desired sample rate for each ADC mainboard is then written to them using the same XferData function. This allows both of the ADC mainboards to begin sending data to the computer. The software then enters a loop that can be terminated by the user pressing any key. During each loop, a data transfer from each ADC mainboard is initiated using the XferData function. This function starts a data transfer from an ADC mainboard and then waits for it to complete. It copies the data transferred from the ADC mainboard into the corresponding allocated memory buffer. Once the data transfer is completed, the memory buffer is transformed into 24 byte data packets, packet numbers are examined for correctness, and written into a text output file for further analysis. If data packet corruption is detected by examining the packet numbers, the loop is automatically terminated. To determine the data transfer rate, each data transfer from the ADC mainboards is timed just like the PCI interface board data transfers were. The loop then repeats until the user presses a key. Once the loop has ended, a 16-bit word consisting of all zeroes is written to both of the ADC mainboards to stop all data transfers and reset the ADC mainboards. Both of the memory buffers are then freed from computer memory using the standard free() C function. Finally, both of the USB devices are closed for data transfers by using the Close function and the program exits.

This chapter highlighted the software tools used to verify the correct operation of the ADC mainboard and PCI interface board FPGA designs. It also provided the tools needed to determine the data transfer performance able to be obtained from both the PCI and USB data interfaces. The following chapter summarizes the individual FPGA design performances such as FPGA logic resources required and maximum FPGA logic clocking rates as well as the software data transfer performances obtained.

CHAPTER 5
DESIGN RESULTS

5.1 FPGA DESIGN RESULTS

Each FPGA design was compiled using the Altera Quartus II software. Default compilation settings provided by the Quartus II software were used. Once compilation was completed, the Quartus II software provided a summary of logic resources needed in the FPGA as well as the maximum clocking frequencies for the FPGA logic. Table 5-1 summarizes the FPGA resources and maximum clocking frequencies for the ADC mainboard design. Table 5-2 lists the design performance obtained for the PCI interface board.

Table 5-1: ADC Mainboard Compilation Results

FPGA Design Item	Used	Available	Percent Used
Logic Resources	2089	5980	34%
Memory Resources	53512	92160	58%
40 MHz Clock	40 MHz	82.45 MHz	49%
105 MHz Clock	105 MHz	335.35 MHz	31%

Table 5-2: PCI Interface Board Compilation Results

FPGA Design Item	Used	Available	Percent Used
Logic Resources	802	5980	13%
Memory Resources	81920	92160	88%
50 MHz Clock	50 MHz	109.29 MHz	46%
75 MHz Clock	75 MHz	127.68 MHz	59%

5.2 VERIFICATION SOFTWARE RESULTS

The serial output data interface and USB 2.0 data interface from the ADC mainboards were independently tested for data transfer rates attainable. The serial output interface using the PCI interface board was tested first. Two ADC mainboards were connected to the PCI interface board prior to running the software. The verification software was setup to program both of the ADC mainboards connected to the PCI interface board to operate at the maximum sample of 156,250 Hz. Using a data packet size of 24 bytes per eight channel bank, a sample rate of 156,250 Hz, and two ADC mainboards, the total expected transfer rate was 2*3*156,250*24 = 22.5 megabytes per second (MBps). The verification software showed a data transfer rate of 22.5 MBps proving that two ADC mainboards could be connected to the PCI interface at the maximum sample rate required and that the PCI interface would not be the bottleneck in the data transfer stream. To determine the absolute maximum data transfer rate of the PCI interface board, a modification to its FPGA design was conducted. 16 bits were internally written to the input SDRAM FIFO at a rate of 75 MHz to provide a data input rate of 150 MBps, well above the maximum rate of the PCI bus of 132 MBps. When the verification software was started it reported a maximum transfer rate of 96.15MBps. This limitation was apparently brought on by the maximum clocking rate of the PLX PCI interface chip of 50 MHz but it was evident that the transfer rate attainable was over four times needed.

The USB interface from two ADC mainboards was tested next. Two ADC mainboards were connected to the computer running the verification software setup to program each of the mainboards for a maximum sample rate of 156,250 Hz. By

referencing the CyApi literature, it was determined that a data transfer block size of 512 bits was required to be able to obtain the maximum transfer rate from the Cypress Semiconductor USB interface chip. This involved using multiple memory pointers to transform the USB data buffers into complete ADC mainboard data packets while keeping track of the bytes that were not used since the data transfer size needed was a non-integral number of ADC mainboard data packets These unused bytes were then stored in memory for use in subsequent ADC mainboard data packet construction. After running the verification software, a maximum data transfer rate of 18.45 MBps was observed. This limitation was attributed to the data transfer latency introduced by the 8-bit data interface provided from the Altera FPGA to the Cypress Semiconductor USB interface chip. The Cypress Semiconductor USB interface chip provided a 16-bit data interface but because of a limited number of I/O pins on the Altera FPGA only 8 data lines were able to be connected. This resulted in a reduced data transfer time and subsequently a reduced data transfer rate. To ensure that data from at least one ADC mainboard running at the maximum sample rate of 156,250 Hz was able to be read into the computer, only one mainboard was connected to the computer and the verification software was setup to program the mainboard for the 156,250 Hz sample rate. After running the software, a data transfer rate of 11.25 MBps was observed. This demonstrated that at least one ADC mainboard operating at the maximum sample rate could be read into the computer. The verification software was then altered to program two ADC mainboards at a sample rate of 78,125 Hz and two ADC mainboards were connected to the computer. The verification software reported a data transfer rate of 11.25 MBps and proved that two mainboards could be operated at half the maximum

sample rate which was quite good but did not meet the goal of two mainboards operating at a sample rate of 156,250 Hz.

After completing all of the verification needed on the FPGA designs by the software program, the software program was provided to a software vendor. Their employees used the verification software as an example to incorporate both of the data interfaces provided by the ADC mainboard and the PCI interface board into their data analysis and display software. Once the software company completed this, a final verification of correct operation from the ADC mainboard and PCI interface board was conducted with successful results.

This chapter reported the results from both of the FPGA designs and also the verification software. Both of the FPGA designs exceeded the required performances. The verification software discovered that the PCI interface outperformed all needed requirements but that USB data transfer scheme needed some substantial modifications to obtain an optimal performance. Once these modifications were completed, however, the USB interface still did not meet the goal of the thesis for two ADC mainboards running at the maximum sample rate of 156,250 Hz. It was verified, however, that two ADC mainboards running at half of the maximum sample rate or one ADC mainboard operating at the full rate could easily be read into a computer using the CyApi driver. Finally a successful integration of the ADC hardware into a commercial software program used by the CADDMAS system was completed.

CHAPTER 6
SUMMARY AND RECOMMENDATIONS

This thesis presented a basic description of the CADDMAS and how it interfaces to its data streams. The original ADC hardware used by CADDMAS was then detailed as well as the shortcomings of its ADC hardware. A list of goals for this thesis was outlined. The thesis then described the new ADC hardware designed for the CADDMAS along with its enhanced capabilities. A detailed description for both of the FPGA designs utilized for the data transfers from the new ADC hardware into the CADDMAS was then presented. A simple verification program to validate the proper operation of the new ADC hardware was then illustrated. The performance results from both of the FPGA designs and the verification software were outlined as well as the difficulties in obtaining the desired maximum transfer rates for the USB interface. The verification software was then provided to a commercial software vendor to incorporate the ADC hardware interfaces into their data analysis and display software. This resulted in a successful integration of the ADC hardware and the commercial software.

In all, both the ADC mainboard and the PCI interface board FPGA designs provided adequate performance necessary to transfer data from one ADC mainboard operating at the maximum sample rate required by the system. The PCI interface board provided a data path that was capable of transferring data from two ADC mainboards operating at the highest sample rate of 156,250 Hz. Only the USB data interface provided by the ADC mainboard did not meet the goals of the thesis. However, two ADC mainboards operating at half the maximum sample rate or one ADC mainboard operating at the full rate were successfully read into the computer using the USB

interface. This was the only observed bottleneck in the data transfer path into the computer. Based upon testing performed and the rates obtained, it should be possible to improve the performance of the USB interface to the rate of two ADCs at 156,250 Hz sample rate by increasing the data path into the USB interface chip from 8 to 16 bits wide. This will require going to an FPGA in a Ball Grid Array package with at least 8 more I/O pins. No other changes to the design should be necessary.

REFERENCES

REFERENCES

[1] "AD7722 – 16-bit 195 kSPS CMOS Sigma-Delta ADC", Data Sheet, Rev B, Analog Devices, Inc.

[2] "FLEX 10K Embedded Programmable Logic Device Family", Data Sheet, ver. 4.2, Altera Corporation, San Jose, CA, January 2003.

[3] "S5935 PCI Product", Data Book, Applied Micro Circuits Corporation, San Diego, CA.

[4] "Cyclone Device Handbook, Volume 1", Data Book, Altera Corporation, San Jose, CA, August 2005.

[5] "CY7C68001 EZ-USB SX2 High Speed USB Interface Device", Data Sheet, Rev H, Cypress Semiconductor Corporation, San Jose, CA, December 2005.

[6] "Cypress CyAPI Programmer's Reference", Manual, version 1.5.11, Cypress Semiconductor Corporation, San Jose, CA, 2003.

[7] "PLX Technology PCI Software Development Kit Programmer's Reference Manual", Manual, version 4.20, PLX Technology, Taiwan, November 2003.

[8] "Quartus II Version 6.0 Handbook", Data Book, Altera Corporation, San Jose, CA, 2005.

[9] Terry W Hayes, Private Communication.

[10] Bruce W Bomar, Private Communication.

VITA

Joshua Brandon Jones was born in Gallatin, Tennessee on April 27, 1980. He was raised in White House, Tennessee and graduated from White House High School in 1998. Brandon entered Tennessee Technological University in Cookeville, Tennessee and received a Bachelors of Science degree in Computer Engineering in 2002. After graduation, Brandon accepted an engineering position with the Information Technology department at Arnold Engineering Development Center. Once there, he continued his studies at the University of Tennessee Space Institute in Tullahoma, Tennessee. In December of 2006, Brandon officially received a Master of Science degree in Electrical Engineering.