To the Graduate Council

I am submitting herewith a thesis written by Brannon Paul Wells entitled "Development of a TIM Module for Data Buffering and Reconfigurable Computing " I have examined the final 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 Bomar, Major Professor

We have read this thesis
and recommend its acceptance

Roy S Joseph

L Montgomery Smith

Accepted for the Council

Associate Vice Chancellor and
Dean of The Graduate School

DEVELOPMENT OF A TIM MODULE FOR DATA BUFFERING AND RECONFIGURABLE COMPUTING

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

Brannon Paul Wells
August 2000

DEDICATION

I would like to dedicate this thesis to my parents, Brad and Beth Wells. Their support and love, coupled with their humor and sense of perspective, have been fundamental to the perseverance of my sanity throughout my life...especially during this project And though they differ, unbelievably so at times, their examples have provided me with the finest of personal goals for my own life, goals that I will forever attempt to achieve.

ACKNOWLEDGEMENTS

I would like to thank Dr Jorgenson for convincing me that there is never a question too stupid to ask, and Dr. Bomar for kindly answering all of them.

This research was supported in part by Sverdrup Technology, Inc under contracts A96B-08-99-06 and A96B-08-00-06 with The University of Tennessee Space Institute (UTSI).

ABSTRACT

The development of a Texas Instruments Module (TIM) compliant memory module is presented The module acts as a high-speed buffer for data acquired from an A/D TIM module and can also be used for reconfigurable computing applications The module provides 256 Mbytes of SDRAM memory (256 times the memory available on current TIM modules with similar functionality), and uses the Altera APEX 20K and FLEX 10KE Programmable Logic Devices (PLDs) for all control operations

The development of the memory module was divided into two parts, the hardware design and the software design The hardware design includes requirements capture, functional description and schematic design, and PCB layout The software design covers the development of the SDRAM controller implemented in the APEX 20K and the communications interface implemented in the FLEX 10KE, which includes the definition of a custom method for 16-bit data transfer between the A/D TIM module and the memory module.

The memory module has been constructed, configured, and tested. The module is capable of storing up to 256 Mbytes of data from an A/D TIM module at a rate of up to 116 Mbytes/sec, and is able to transfer all of the stored data to a PC via the TIM-40 PCI Motherboard to which it and the A/D TIM module are mounted

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Project Overview

The processing of high-bandwidth digital media, such as digitally encoded video, requires the highest speed digital signal processors. In some cases, even the most current DSP technology is not fast enough to process this data at the speeds required One option to overcome this problem is to create a large, high-speed buffer The data can be stored in this buffer, and then fed to the signal processor at a rate that it can handle

Current technology has also made a decisive move towards reconfigurable designs With the introduction of faster and faster generic, reconfigurable processors, even the fastest applications are making this move. Reconfigurable designs provide the ability to use a single design for multiple applications, saving development time and resources One type of reconfigurable processor is the Programmable Logic Device, or PLD

The following presents the development of a high-speed buffer module that conforms to the TIM-40 module specifications (referred to as a TIM module) The primary purpose of this module is to store data received from an analog to digital (A/D) converter module at a rate of 75 MB/sec, and then output this data to a computer via the TIM-40 PCI motherboard to which the modules are mounted Figure 1-1 illustrates this primary application for the high-speed buffer module

The module provides 256 Mbytes of SDRAM memory for data storage and uses the Altera APEX 20K and FLEX 10KE Programmable Logic Devices for memory and communications control Though the module's primary purpose is to serve as a high-speed buffer, its use of PLDs and its large amount of available memory make it well equipped to perform in a wide variety of other applications. One such immediate application is as part of a reconfigurable computing network

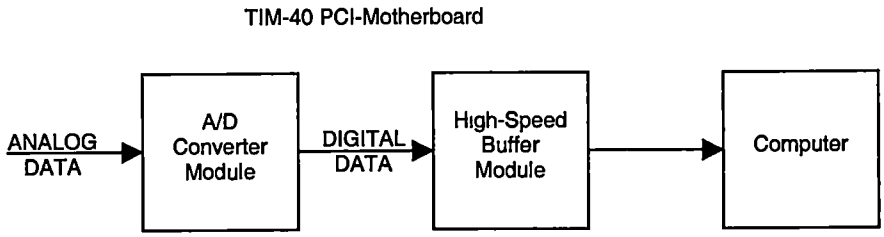

Figure 1-1: Primary Application for the High-Speed Buffer Module

1.2 Design Requirements

The SDRAM TIM Module was designed in order to meet the following set of requirements

1 The module must be able to store 256 MB of information from an A/D module at a rate of 75 MB/sec

2 The module must comply with the TIM-40 module specifications, as specified in Section 2.3.

3 The module must provide sufficient adaptability to be used as part of a reconfigurable computing system

1.3 Thesis Outline

Chapter 2 presents background information necessary for the development of the module, including information pertaining to the TIM-40 Module Specifications, the C40 Communications Protocol and Interface, SDRAM, and Altera APEX Devices Chapter 3 discusses the process used for the development of the hardware portion of this design. Covered topics include all portions of the design process, from requirements capture and part selection, to front-end and back-end design. The development of the configuration programs for the

programmable logic devices and the definition of a custom method for 16-bit data transfer between the A/D TIM module and the memory module is presented in Chapter 4 Chapter 5 describes each of the tests that were performed on the device after construction as well as the results that were obtained from the execution of these tests. Conclusions reached from this design process as well as future recommendations for further development of the module are presented in Chapter 6.

Chapter 2

Background

2.1 TIM Module and TIM-40 Specifications

A TIM Module is a daughter board that conforms to the Texas Instruments Module (TIM-40) Specifications These specifications were devised in order to create a "cost effective parallel processing module standard" [1]. The module can serve in a wide variety of functions, from parallel computing to image and signal processing

The TIM module is supported by a motherboard, which provides the operating environment for the module This operating environment includes the power, communication paths (such as from module to module, module to PC, etc), and signals such as the clock, reset, and JTAG [8] lines PCI, VME, along with many other backplanes are currently supported

The TIM-40 Specifications provide physical and electrical constraints for the TIM modules The following sections present those physical and electrical specifications that were important to the current design.

2.1.1 Physical Specifications

The TIM-40 Standard dictates the dimensions that the module must conform to It also provides specifications for the connectors provided by the module for use with the motherboard Figure 2-1 illustrates a single-width TIM module The module must be 4 2 inches in length and 2 5 inches in width The components on the TIM module must also not exceed maximum height requirements when mounted on the module, so that no contact between module components and motherboard components and/or motherboard will occur

The TIM module provides two primary connectors for connection with the motherboard, labeled on Figure 2-1 as J1 and J2 This connection provides 4 communications ports, each 12-

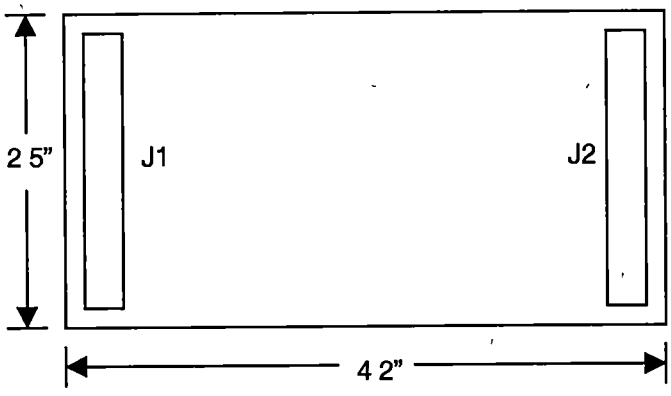

Figure 2-1: Single-Width TIM Module

bits wide, clock and reset signals, JTAG signals, and power and ground connections. The connector used for J1 and J2 is the Hirose FX-4 Series

2.1.3 Electrical Specifications

The TIM-40 specifications describe the power supplies provided to the TIM module by the motherboard The motherboard provides a +5V, -12V, and +12V supply, as well as ground connections Also described are the communications standards for the module TIM modules communicate with each other and the motherboard via 4, 12-bit wide bidirectional communication ports Additional signals, such as RESET, CLOCK, and JTAG lines are also provided, and the use of these lines is specified

2.2 C40 Communications Protocol and Interface

The C40 Communications Protocol defines the method in which data is transferred between the various TIM modules on a TIM-40 motherboard, and between the TIM modules and the motherboard itself Communication takes place on one of the available 12-bit wide

communications ports All twelve of the bi-directional lines are implemented, eight lines for data (d[7 0]), and four lines for handshaking (CACK, CRDY, CREQ, CSTRB)

A C40 Communications Interface for implementation within PLDs is available and provides the processes necessary to communicate using the C40 Communications Protocol. Figure 2-2 illustrates the C40 communications interface.

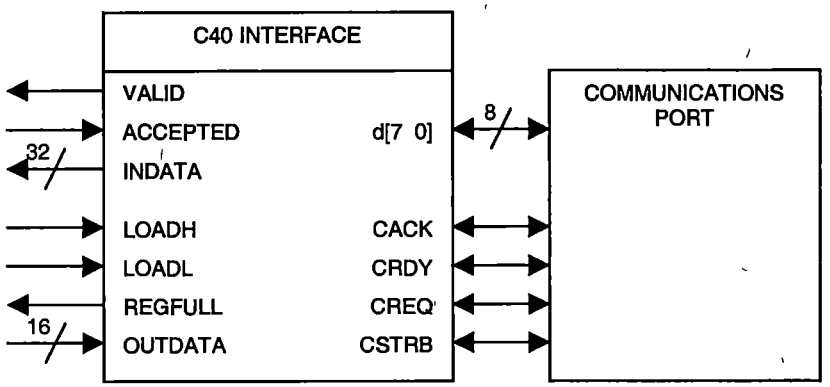

Figure 2-2: C40 Communications Interface Diagram

The following lists the procedure for sending data via the interface:

1. The user determines if the REGFULL line is at a logic "0" This verifies that the interface is ready to receive data
2. The user places the first 16 bits of data to be sent on the OUTDATA 16-bit wide bus. The user then places a logic "1" on the LOADL line This tells the interface to load the data currently on the OUTDATA bus into the interface.
3. The user places the second 16 bits of data to be sent on the OUTDATA 16-bit wide bus The user then places a logic "1" on the LOADH line This tells the interface to load the data currently on the OUTDATA bus into the interface
4. The REGFULL line goes to a logic "1" and remains at a logic "1" until the 32-bit work has been transferred over the C40 communications port.

The following lists the procedure for receiving data via the interface·

1. The user determines if the VALID line is at a logic "1". This tells the user that the interface has received data via a C40 communications protocol compliant communications port
2. The user reads the values present on the INDATA 32-bit wide bus.
3 The user places a logic "1" on the ACCEPTED line to tell the interface that the data has been read

2.3 SDRAM

SDRAM (Synchronous Dynamic Random Access Memory) represents industry's current answer to the consumer need for high-speed, high-density memory elements A further development upon DRAM technology, SDRAM is DRAM in which all commands and data transfers are synchronized to an input system clock. Figure 2-3 illustrates the functional block diagram of an SDRAM element [2]

2.3.1 Operations

SDRAM chips are accessed and controlled via the data bus, address bus, and control lines. The data bus is 8 bits wide, referred to as DQi in Figure 2-2 ADD is the address bus, composed of 12 address bits, and 2 bank select bits. Memory is organized into 4 banks, each of which is broken into 4096 pages. Each page is selected by a row address and is then further divided into 1024 8-bit bytes using a column address Bytes in memory are accessed by first passing the SDRAM the bank and row address, and then the column address SDRAM control lines include the clock (CLK), chip select (CS), row-access strobe (RAS), column-address strobe (CAS), write enable (WE), clock enable (CKE), and data mask (DQM). For this design, the

FUNCTIONAL BLOCK DIAGRAM

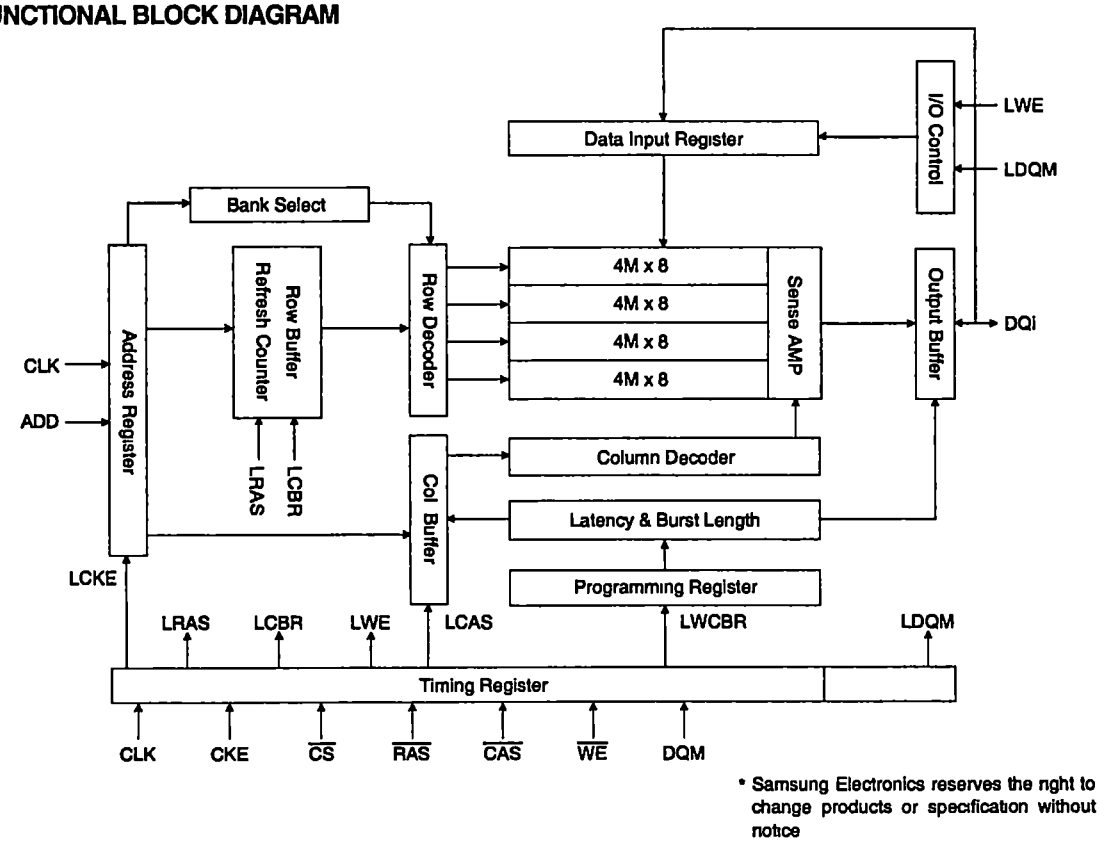

Figure 2-3: SDRAM Functional Block Diagram

control lines are organized and referred to as an array, RAMCNTL(4 .0), organized as follows CKE, DQM, RAS, CAS, WE, respectively. Commands are passed to the SDRAM device via this bus The commands and their functions will now be discussed

Row Active

The row active command is given while the SDRAM is in the idle state. The bus and row address of the memory to be accessed are present on the address and bank bus respectively when this command is given [3] After a designated row active time, the row is ready to be accessed. Table 2-1 presents the RAMCNTL state necessary for the row active command.

Table 2-1: RAMCNTL Command States

ACTION	CKE	DQM	RAS	CAS	WE
Row Active	1	0	0	1	1
Read	1	0	1	0	1
Write	1	0	1	0	0
Precharge	1	0	0	1	0
Auto-refresh	1	0	0	0	1

Read

The read command is given while the SDRAM is in the row active state after the row active time is satisfied The column address of the memory to be accessed is present on the address bus when this command is given Dependent upon the burst length designated in the Mode Register, the SDRAM will begin a read of 1, 2, 4, or 8 bytes, or a full-page length (1024 bytes) after the CAS delay Upon completion of the read, the device is given a precharge command in order to return to the idle state Table 2-1 presents the RAMCNTL state necessary for a read command Figure 2-4 illustrates the timing diagram for a row active command and a full-page length read [4]

Write

The write command is given while the SDRAM is in the row active state after the row active time is satisfied The column address of the memory to be accessed is present on the address bus when this command is given Dependent upon the burst length designated in the

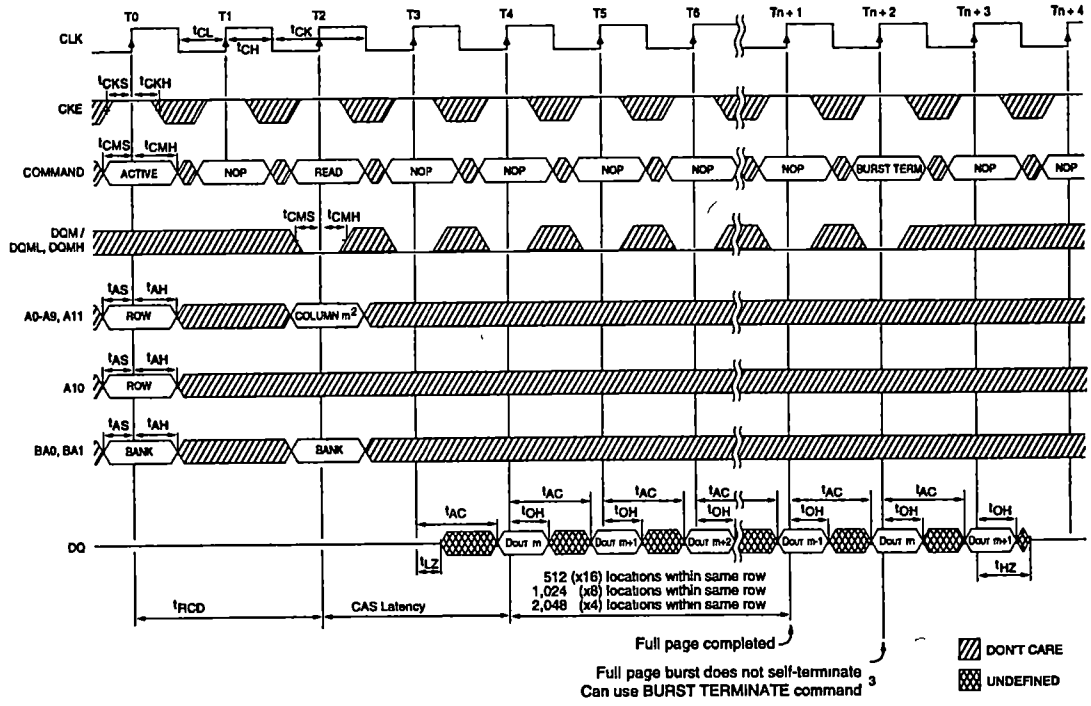

Figure 2-4: Row Active and Full-Page Read Command Timing Diagram

Mode Register, the SDRAM will begin a write of 1, 2, 4, or 8 bytes, or a full-page length of data present on the RAMDATA bus. Upon completion of the write, the device is given a precharge command in order to return to the idle state Table 2-1 contains the RAMCNTL state necessary for a write command Figure 2-5 illustrates the timing diagram for a full-page length write

Precharge

A periodic precharge command is necessary for accurate reading of the RAMDATA bus, and is one of the methods of returning to the idle state from a read or write operation After the precharge time is satisfied, the SDRAM device returns to the idle state. Table 2-1 contains the

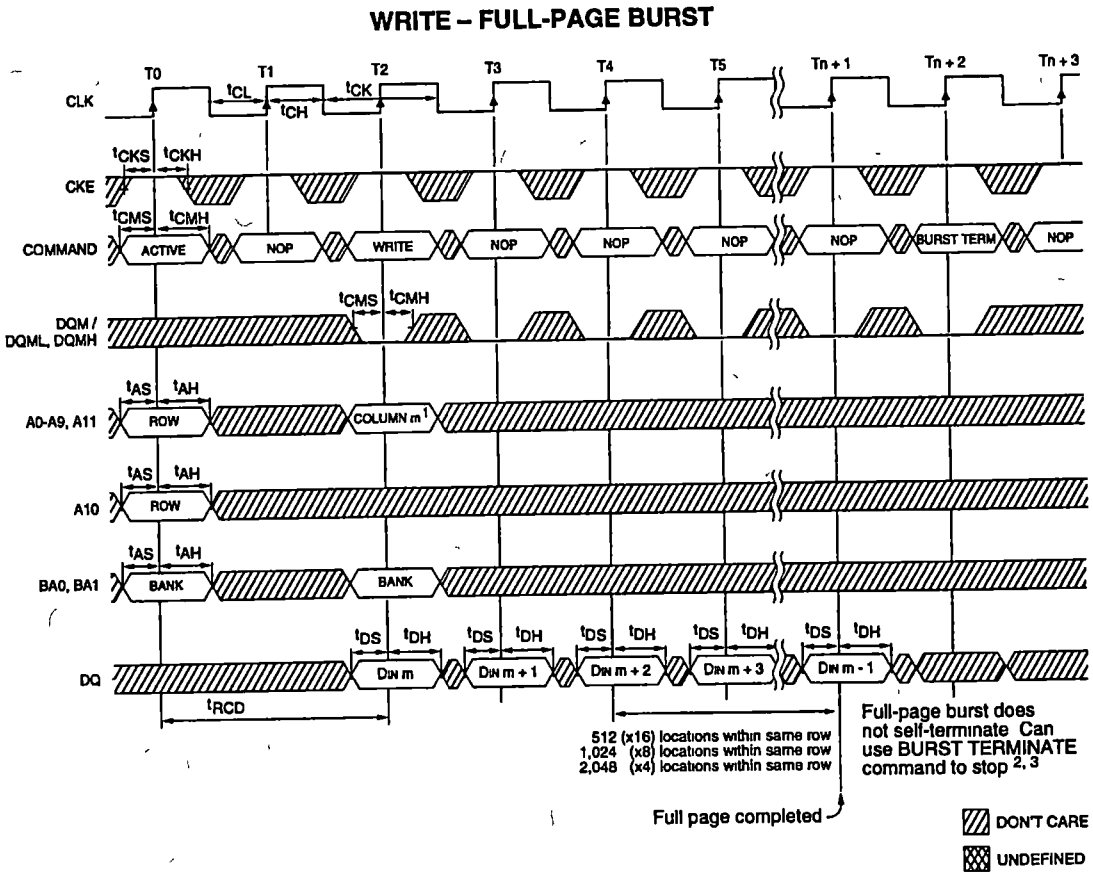

Figure 2-5: Row Active and Full-Page Write Command Timing Diagram

RAMCNTL state necessary for a precharge command Figure 2-6 illustrates the timing diagram for a normal read and write precharge [5].

Refresh

In order to maintain the current state of a memory cell and avoid corruption, all rows of each SDRAM device must be refreshed every 64 ms. The most commonly used refresh command is the autorefresh command This command is issued from the idle state, at which

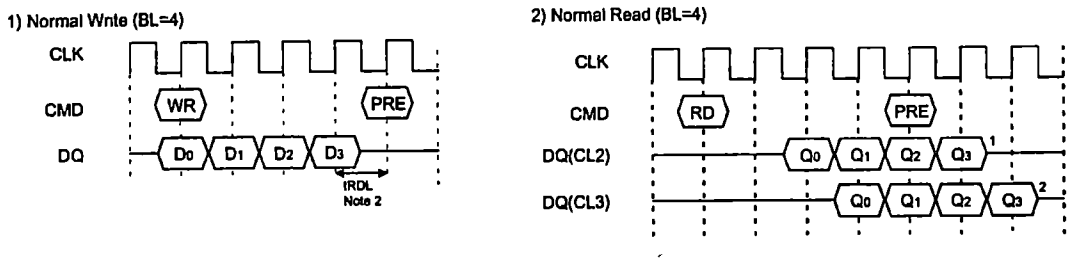

Figure 2-6: Precharge Command Timing Diagram

time the device refreshes an entire row in parallel An internal counter is then incremented. In this way the device keeps track of which row needs to be accessed next The user does not need to keep track of the row address to be refreshed, the only requirement is that each row is refreshed every 64 ms Table 2-1 contains the RAMCNTL state necessary for an autorefresh command Figure 2-7 illustrates the timing diagram for an autorefresh

2.3.2 Advantages and Disadvantages

Just as any other device, SDRAM chips present the user with advantages and disadvantages to be considered One SDRAM advantage is the high-density of the memory inherent with the single transistor memory cell design [6] However, this memory cell design has its disadvantages, such as the necessity for a refresh cycle Also, due to the decreased size of each memory element with increased densities, capacitance is lowered, and therefore the cell is more susceptible to noise and possible data corruption SDRAM is less susceptible to noise present on the data, access, and control lines, however, since the state of these lines is only read at the rising edge of the input system clock.

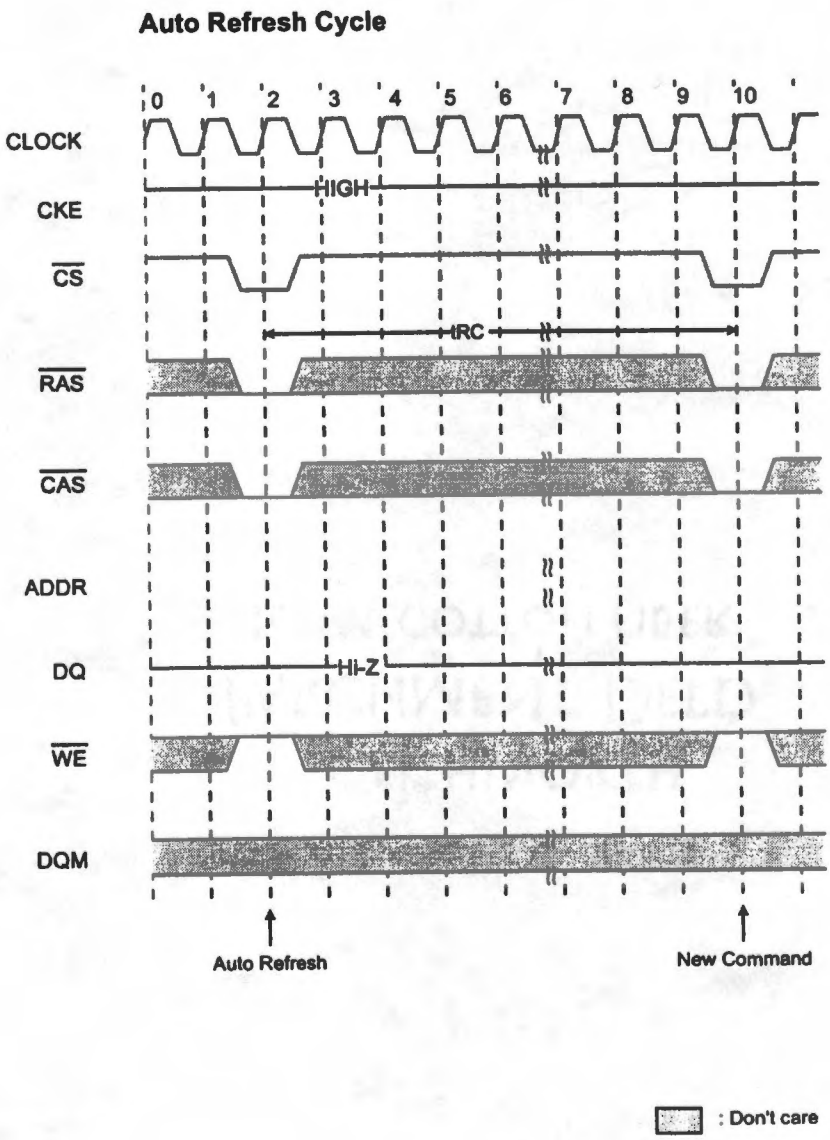

Figure 2-7: Autorefresh Command Timing Diagram

2.4 Altera APEX Programmable Logic Devices

The Altera APEX20K100 is a programmable logic device [7]. Upon power-up, a configuration program is loaded into the APEX20K100, causing the APEX device to behave in the desired fashion. Configuration is lost once power is removed, so the device must be configured, not necessarily with the same configuration program, each time the device is turned on. The APEX device is said to be fully reconfigurable and reprogrammable

2.4.1 Physical Construction

APEX 20K devices contain LEs (logic elements), ESBs (embedded system blocks), and FastTrack Interconnect. LEs contain a 4-input look-up table, a programmable register, and a carry and cascade chain Figure 2-8 shows an APEX 20K LE Every 10 LEs form a logic array block (LAB). ESBs are used to implement memory elements, as well as product-term logic. MegaLAB structures, which make up an APEX 20K device, are made up of 16 LABs and an ESB, along with FastTrack interconnect FastTrack interconnect provides the interconnection between and within the MegaLAB structures

2.4.1 Configuration

APEX 20K devices can be configured in a number of ways. Configuration can occur serially or through an 8-bit wide bus Memory elements, such as flash-memory, are often used to provide configuration information. Bit serial EEPROM configuration devices, such as the Altera

Figure 5. APEX 20K Logic Element

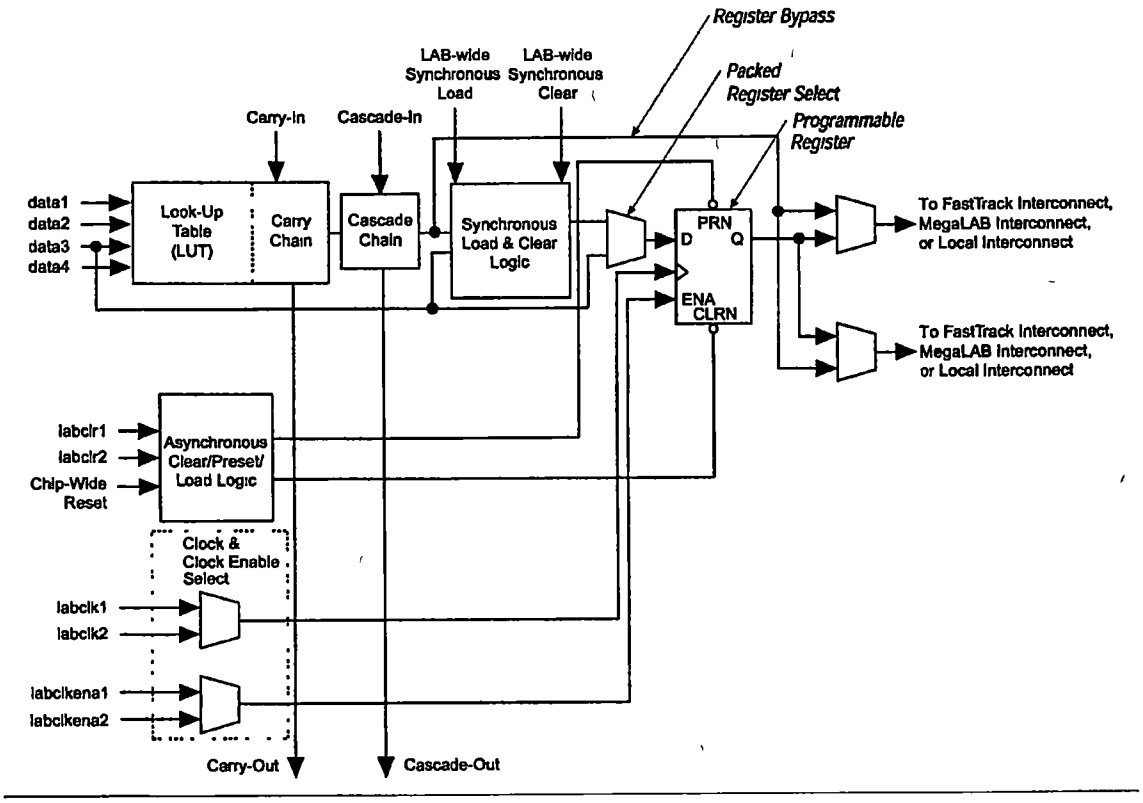

Figure 2-8: APEX 20K Logic Element

EPC2, can also be used to configure the APEX 20K The JTAG boundary-scan test (BST) circuitry can also be used to configure the device, though this circuitry is normally used to test its integrity [8]

2.4.2 Altera FLEX Programmable Logic Devices

An Altera FLEX 10KE programmable logic device was also used in this design The FLEX 10KE is a programmable logic device similar to the APEX 20K, but has 5V tolerant I/O and is less dense in comparison to the APEX 20K. In terms of its use and construction, however, it is virtually identical to the APEX 20K.

2.4.3 Advantages and Disadvantages

The use of PLDs in any design provides advantages as well as disadvantages The most prominent among the advantages of using a PLD is its ability to be reconfigured an infinite number of times (within the life of the device) This greatly simplifies the hardware design, placing a greater dependence of the operations of the product on the software implemented within the PLD It also allows general-purpose devices to be designed, which can be configured for their specific use, whatever that might be And since Altera PLDs provide multiple configuration options, the initial investment for configuration capabilities is minimal. However, the need to configure the device each time power is applied often requires the addition of a configuration device and/or circuit.

Chapter 3

Hardware Design Description

3.1 Requirements Capture

The hardware portion of this project was designed in order to meet the following requirements

1. The module must conform to the TIM-40 Module Specifications
2. The module must provide 256 MB of storage capacity
3. The module must provide control circuitry for reading and writing to the storage elements.
4. The module must provide JTAG interface from an onboard header and through the J1 TIM connector
5. The module must provide a method for altering the on-board clock frequency
6. The module must provide control circuitry for the interface between the module and the communications ports.
7. The module must be constructed in a manner that provides sufficient noise immunity and minimal noise production
8. The module must provide a high level of reconfigurability for potential reconfigurable computing applications
9. The module must be capable of a 75MB/sec storage rate
10. The module must provide multiple configuration options for potential reconfigurable computing applications

3.2 Part Selection

Samsung SDRAM

In order to meet design requirements 2 and 9, it was necessary to find a high-speed, high-density memory element(s) for the project. SDRAM typically can be clocked at a rate of 100MHz, exceeding the rate of data storage necessary for the design. At the time of design, Samsung Semiconductors was one of the few companies that produced a 16MB SDRAM device Micron also produced a similar device, but it was not available to the consumer at the time of design. Due to the pin compatibility of both devices, Micron chips may be used in the future as an alternative to the Samsung SDRAM device. Packaging of the device also conformed with all TIM-40 module component dimension specification considerations

Altera APEX20K100

The control circuitry for the SDRAM had to be very fast and reconfigurable. Due to these requirements, a high-speed programmable logic device was the obvious choice. Altera's newest line of PLDs, the APEX devices, provided a high amount of gates and memory. The APEX devices are 2 5V products, however, they are 3.3V I/O tolerant, making them voltage compatible with the 3 3V SDRAM chips Unfortunately, they are not 5V tolerant and therefore cannot be used to connect to the 5V TIM Module communication ports The TQFP 144-pin package for the APEX device also complied with all TIM-40 module component dimension specifications

Altera FLEX10K50E

Since the communications ports of the TIM-40 module specification "speak" in 5V, the communications port controller circuitry must also be I/O operational at the same level. In order

to achieve this, as well as comply to the need for a high level of reconfigurability, the FLEX10K50E was chosen to provide control of the communications interface. The TQFP 144-pin package for the FLEX device also complied with all TIM-40 module component dimension specifications.

Others

In order to provide the features described in requirements 4, 5 and 10, additional devices were used in addition to those supporting the control circuitry and SDRAM.

The Cypress ICD2053B programmable clock generator was used to provide the variable clock mentioned in requirement 4 Given an input clock, the ICD2053B is capable of outputting a wide range of frequencies as determined by a binary code clocked into the device.

In order to allow the JTAG chain to be interfaced by either the on-board header or the J1 connector, it was necessary to implement a switch The Pericom PI5C3257 MUX/DEMUX was used in conjunction with an on-board jumper in order to accomplish this goal

The design is required to provide multiple configuration options The JTAG header provides one such option Another configuration method is using an Altera configuration device The Altera EPC2 is such a device, and was chosen for use with the FLEX10K50E The Atmel AT29LV020 Flash Memory was chosen to configure the APEX20K100 outside of the JTAG chain.

3.3 Functional Block Diagram and Schematic Design

Figure 3-1 illustrates the interaction of the components of the design The following further describes the major functionality of these components when working in conjunction with each other

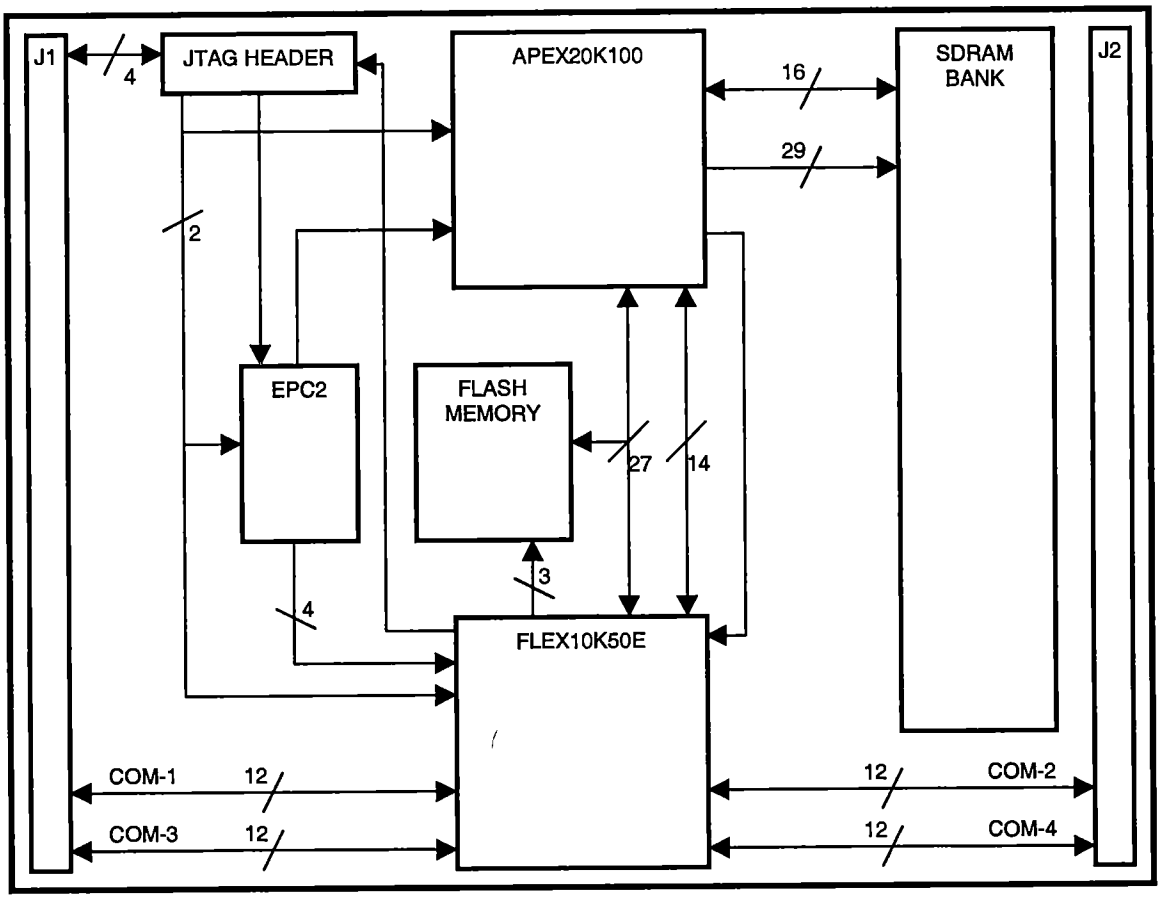

Figure 3-1: Module Functional Block Diagram

The APEX device is connected to the data, address, and control lines of the SDRAM chips All data to and from the SDRAM will pass through APEX device, thereby guaranteeing that the APEX device will have absolute control over the SDRAM.

As noted in Part Selection, the APEX device is not capable of handling a 5V I/O, therefore all information to and from the communications ports must pass through the FLEX device The APEX and FLEX device share a total of 41 lines, 14 of which are dedicated for communication between the two devices, and 27 which are also used for the data and address lines of the flash memory.

The JTAG chain, accessed via the JTAG header or the J1 connector, provides a method of configuring the PLDs and the EPC2 configuration device Under initial testing, the JTAG chain was the method in which the PLDs were configured The JTAG chain is composed of four lines, TCLK, TMS, TDI, and TDO [8] This chain can also be used to test the integrity of the chips included in the chain The TCLK and TMS are parallel lines, connecting to all of the devices included in the JTAG chain the EPC2, the APEX device, and the FLEX device The TDO and TDI lines pass serially from device to device. The JTAG chain order is EPC2, APEX 20K, FLEX 10KE Figure 3-2 illustrates the JTAG connections for the module In order to guarantee that 5V signals are not placed upon any of the APEX I/O pins, 3.3V voltage buffers are included on the TCLK and TMS lines

The EPC2 and Flash Memory are used to configure the PLDs under normal operation The EPC2 configures the FLEX 10K50E, the APEX 20K100 receives its configuration from the Flash Memory. The EPC2 is initially programmed via the JTAG chain The FLEX10K50E is used to program the Flash Memory with code that it receives from the JTAG chain. Due to the nature of the devices, they will retain their programming after power is removed, and must only be reprogrammed when there is a code change

The schematic design for this project was created using the HiWire II PCB design package After the schematic was created, the design package produced a netlist from the schematic that could be checked against the PCB layout in order to confirm that the layout was correct

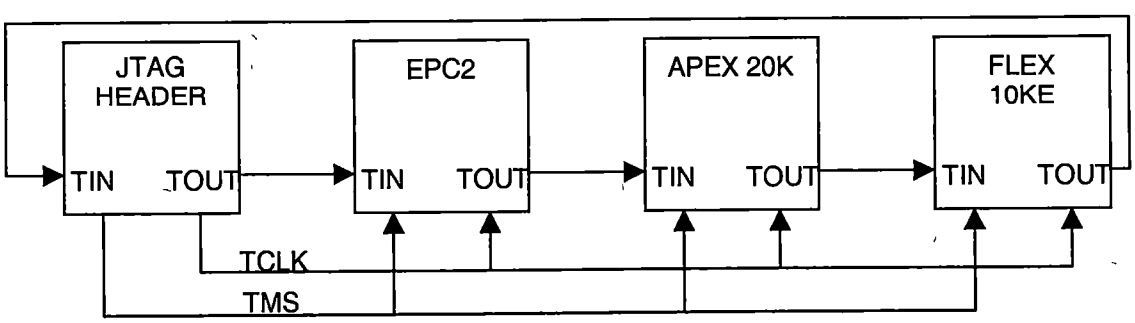

Figure 3-2: JTAG Chain Component Connection Diagram

3.4 PCB Layout

The PCB layout was also created using the HIWIRE II design package It was determined that a 6 layer PCB was sufficient to provide the necessary design space for the project Figure 3-3 illustrates the function of each of these layers. Note that signal layers are separated by either a ground or power plane or the thick material layer found between layers 5 and 6 This layout provides return paths that are very close to the signal paths, thereby reducing inductive loops and radiated noise [9]

The PCB layout was checked against the schematic, and was found to be identical.

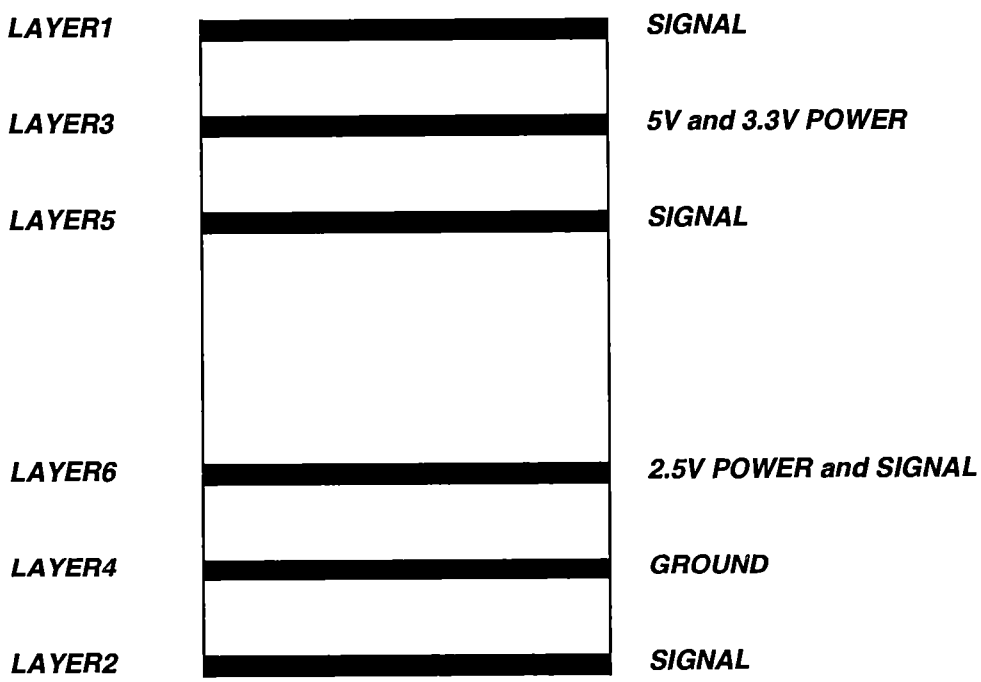

Figure 3-3: PCB Layer Function and Organization Diagram

Chapter 4

Software Design Description

4.1 Requirements Capture

The software portion of this project was designed in order to meet the following requirements

1 The design must provide SDRAM control

2 The design must provide control for the interface between the SDRAM and communications ports

3 The design must control the following processes reading data from an A/D TIM module, storing that data, and sending that data to a PC

4 The design must meet all before-mentioned timing requirements.

4.2 Functional Design and Block Diagrams

The software design was divided into two parts the SDRAM controller implemented in the APEX 20K device, and the communications interface implemented in the FLEX 10KE device Figure 4-1 illustrates the connections between the APEX 20K, the FLEX 10KE, the SDRAM, the A/D TIM Module from which the SDRAM TIM Module receives its data, and the personal computer that the SDRAM TIM Module sends its data to The following sections will first describe the development of the SDRAM controller, and then the communications interface implemented in the FLEX 10KE

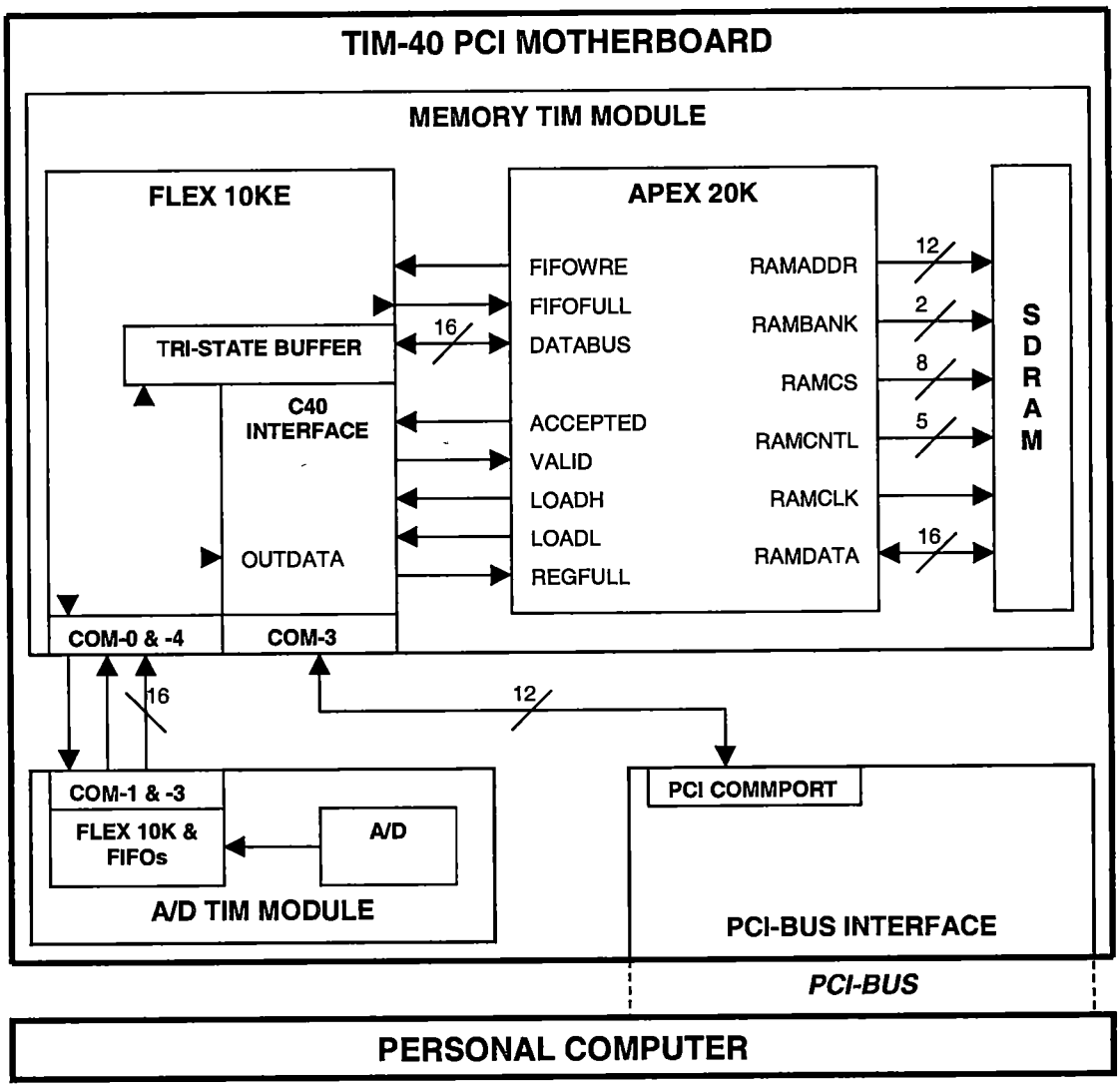

Figure 4-1: Device Connection Diagram

4.2.1 Communications Interface

The FLEX 10KE device was configured as a communications interface The communications interface consists of a C40 protocol interface, a tri-state buffer, and a few pass-through bits (see Figure 4-1)

The pre-constructed C40 Communications Interface was used for the control of communications between the TIM-40 module and the TIM-40 PCI Motherboard. Due to the necessity to transfer data from the A/D TIM module to the Memory TIM module at a rate of 75 Mbytes/sec, the standard C40 protocol, which only supports transfer rates of 20 Mbytes/sec, could not be used Therefore it was necessary to define a custom method for data transfer between the A/D TIM module and the Memory TIM module

Communication between the A/D TIM Module and the Memory TIM Module takes place simultaneously over two communications ports Sixteen of the available 24 lines are used for data transfer from the A/D TIM module to the Memory TIM module. By using 16 lines for data transfer, two bytes of data can be moved at every clock cycle Therefore, in order to satisfy the design requirements, the communications interface (and SDRAM controller) must be able to perform at a minimum clock rate of 37 5 MHz Because data is moving in only one direction, only two lines, FIFOFULL and FIFOWRE, are necessary for data transfer control FIFOFULL is connected to the almost full flag on the 32K deep FIFOs, into which data is stored onboard the A/D TIM Module When FIFOFULL goes to a logic "1", it denotes that almost 32K worth of data is available to be read from the FIFOs to the SDRAM. This permits an entire page (1K) in SDRAM to be written as one block The FIFOWRE line is connected to the output enable pin on the FIFOs One line is used to provide the FIFOs with an output clock signal The other five lines are not used

4.2.2 SDRAM Controller

The SDRAM controller is responsible for maintaining the proper operation of the SDRAM memory. It must also be able to transfer data from the A/D TIM module to the SDRAM and from the SDRAM to the C40 interface onboard the FLEX 10KE device. The operation of the SDRAM controller is divided into four major operations: STARTUP, IDLE, WRITE, and READ. Figure 4-2 illustrates the operational diagram of the SDRAM controller.

***STARTUP** Operation*

The STARTUP operation places the SDRAM in an operational state after power has been applied to the SDRAM TIM module. All states with the prefix "S" produce the waveforms necessary after power-up in order to place the SDRAM in an operational state. Figure 4-3 illustrates the state-machine for the STARTUP operation.

***S0**-Startup0* - Creates an 8000-cycle idle-state. All chips are disabled.

***S1**-Startup1* - All chips are given an all-banks precharge command.

***S2**-Startup2* - Creates a 1-cycle idle-state in order to wait for completion of the precharge.

***S3**-Startup3* – The Mode Register is set for full-page burst sequential read and write access.

***IDLE** Operation*

The idle operation performs multiple autorefresh routines and waits for a command from the controller PC, at which time it waits for the FIFOs onboard the A/D TIM to be almost full. A minimum of two autorefresh cycles is performed each time the idle process is entered, thereby satisfying the autorefresh frequency requirements. Figure 4-4 illustrates the state-machine for the IDLE operation.

***I0**-Idle0* - Creates a 1-cycle idle-state

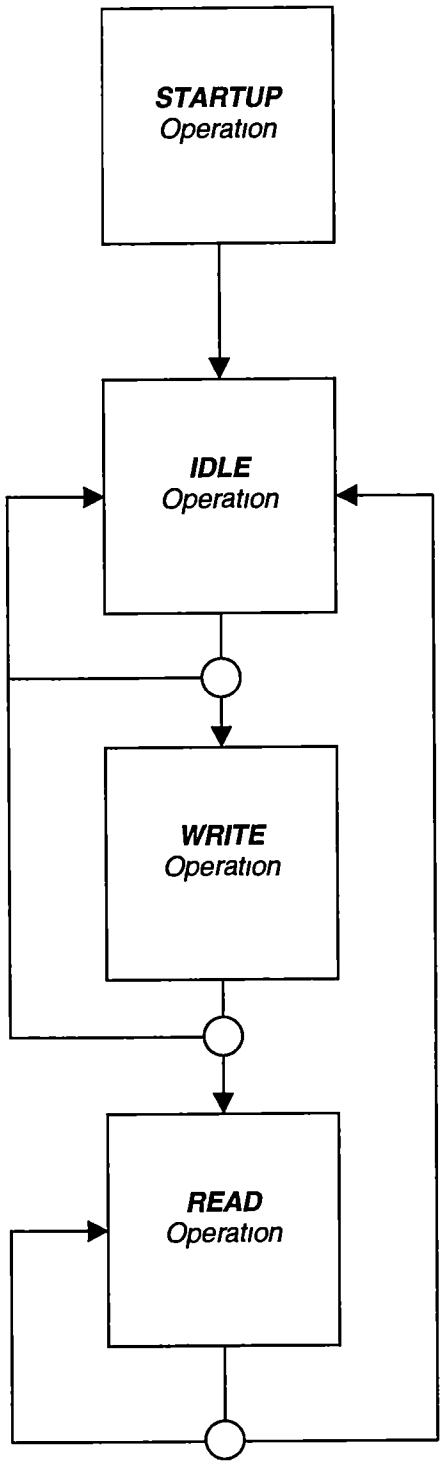

Figure 4-2: SDRAM Controller Operational Diagram

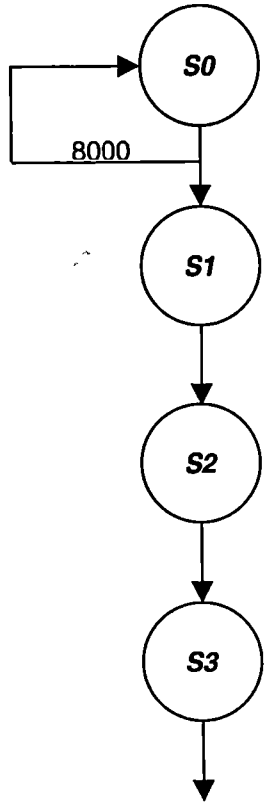

Figure 4-3: STARTUP Operation State-Machine Diagram

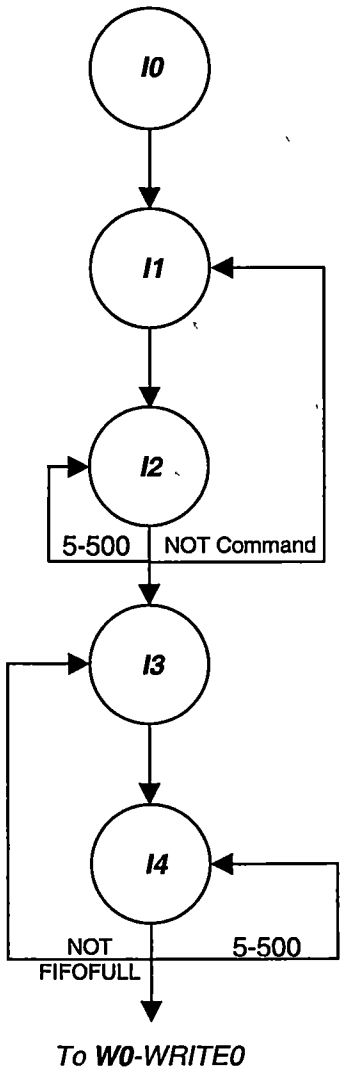

Figure 4-4: IDLE Operation State-Machine Diagram

I1-Idle1 - All chips are given the autorefresh command.

I2-Idle2 - Creates a 5 to 500-cycle idle-state in order to wait for the completion of the autorefresh and a command from the PC. The state checks for a command from the controller PC, denoted by a logic "1" present on the input **VALID** line. If a command is received, it is acknowledged by loading the output line **ACCEPTED** with a logic "1" and the register **command_int** is set. After waiting for completion of the autorefresh, the register **command_int** is checked in order to determine if a command has been received by the controller PC. This process continues for up to 500 cycles, after which the state machine will return to I1 in order to perform another autorefresh. If a command has been received the state-machine increments to I3. The state-machine will continue this refresh - wait and check routine until a command is received from the controller PC.

I3-Idle3 - All chips are given the autorefresh command.

I4-Idle4 - Creates a 5 to 500-cycle idle-state. This state functions almost identically to state I4. It first waits for the autorefresh to complete, and then waits up to 500 clock cycles for the **FIFOFULL** line to go to a logic "1", at which time the state machine increments to W0 Otherwise the machine jumps back to I3.

***WRITE** Operation*

The write operation controls the transferring of one page (1024 16-bit words) of information from the 32K deep FIFOs onboard the A/D TIM module to the SDRAM As long as the memory is not yet full, the write operation exits to the idle process as soon as the write is complete. If the memory is full, the process exits to the read operation upon completion of the page write Figure 4-5 illustrates the state-machine for the WRITE operation.

W0-Write0 - The Active Row Command is loaded onto the **RAMCNTL** bus, with the row address of the page to be written to present on the address bus, **RAMADDR**.

W1-Write1 - Creates a 1-cycle idle-state in order to wait for the row to go active

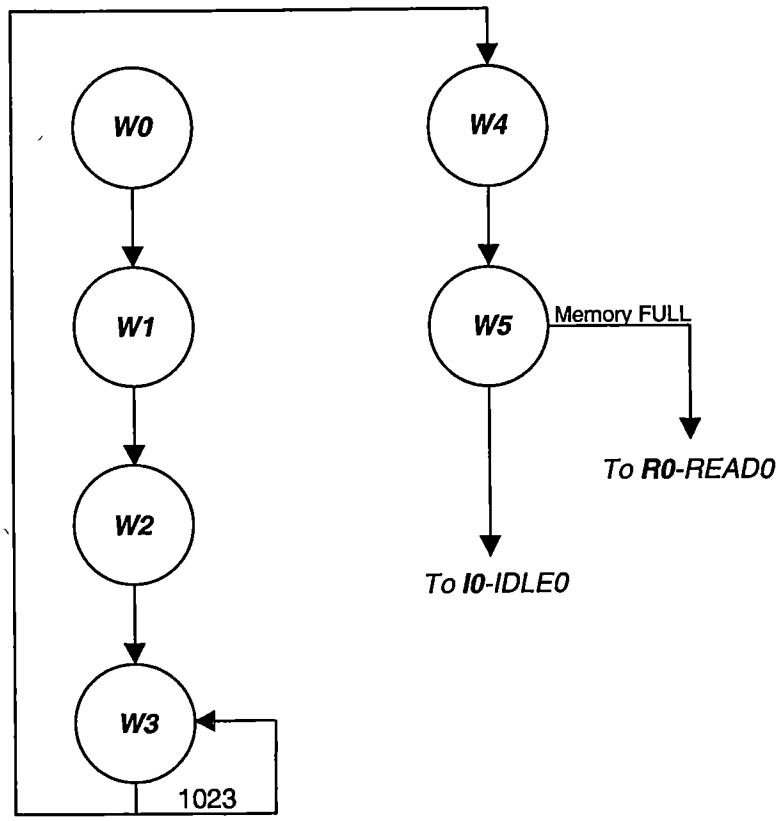

Figure 4-5: WRITE Operation State-Machine Diagram

W2-Write2 - The Write Command is loaded onto the **RAMCNTL** bus The A/D TIM is also notified to begin sending data by placing a logic "1" on the **FIFOWRE** line

W3-Write3 - Creates a 1023-cycle idle-state in order to wait for a page (1024 16-bit words) to be written to memory Row, bank, and chip addresses are also incremented as necessary

W4-Write4 - Creates a 1-cycle idle-state in order to wait for the end of the page-length burst.

W5-Write5 - The Precharge command is given. If it is determined that all chips are full, the state-machine increments states to R0 Otherwise, the machine jumps back to I0 in order to autorefresh the memory and wait until the FIFO is again almost full

***READ** Operation*

The read operation controls the transferring of the entire contents of memory from the SDRAM to the controller PC 32-bits are sent, at which time the process waits for confirmation that the data has been received, and then sends another 32-bit block Data transfer continues until all SDRAM chips have been fully read. The process then exits to the idle process Figure 4-6 illustrates the state machine for the READ operation

R0-Read0 - Creates a 1-cycle idle-state in order to wait for completion of the precharge

R1-Read1 - Sets the Mode Register for all of the SDRAM chips Block size is set to 2 bytes.

R2-Read2 - Creates a 1-cycle idle-state in order to wait for completion of the mode-register set. Also checks if the entire memory system has been transferred If it has, the machine returns to S2 Otherwise, the state machine goes to R3

R3-Read3 - All chips are given the autorefresh command

R4-Read4 - A 5-cycle idle-state is created in order to wait for the completion of the autorefresh.

R5-Read5 - The Active Row command is given, with the row address of the page to be read from present on the address bus.

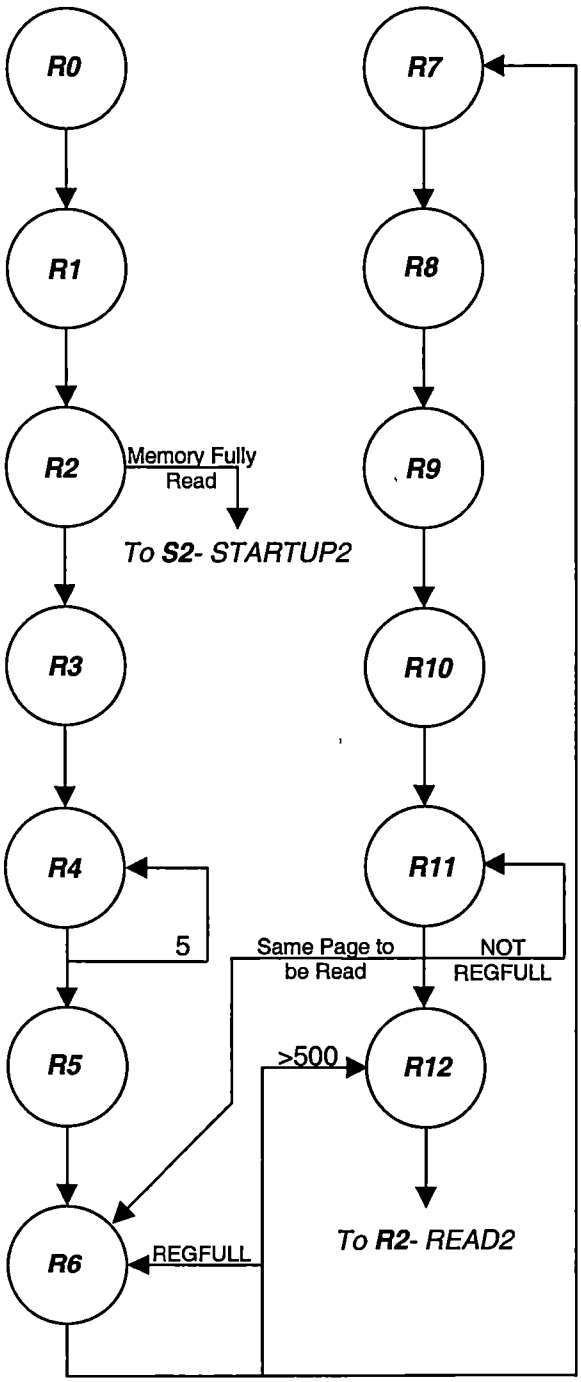

Figure 4-6: READ Operation State-Machine Diagram

R6-Read6 - Creates a 1-cycle idle-state in order to wait for the row to go active The state then waits up to 500 cycles for **REGFULL** to be low, denoting that the communications interface is ready to receive data The state machine then moves to R7 If the 500 cycle count is exceeded, the machine jumps to R12 in order to perform another autorefresh.

R7-Read7 – The Read Command is loaded onto the **RAMCNTL** bus

R8-Read8 - Creates a 1-cycle idle-state in order to wait for the CAS delay. The column address is also incremented

R9-Read9 - The communications interface (FLEX10K50E) is notified that data is present by placing a logic "1" on the **LOADH** line This data constitutes the upper half of the 32-bit word to be sent to the controller PC The column address is also incremented

R10-Read10 - The communications interface (FLEX10K50E) is notified that data is present by placing a logic "1" on the **LOADL** line This data constitutes the lower half of the 32-bit word to be sent to the controller PC

R11-Read11 – The state waits for the **REGFULL** line to go high, confirming that the controller PC has received the data It is then determined if a new page of memory will be read during the next read cycle If so, the state machine goes to R12 Otherwise, the machine returns to R6.

R12-Read12 – The Precharge command is given The state machine then returns to R2

4.3 VHDL and GDF Construction

VHDL was used to implement the state-machines involved in the SDRAM controller code, as well as small individual components within both designs. Graphic Design Files (GDFs) were then used to connect these virtual devices together, in order to create each of the design entities.

VHDL Design

The design of the state-machines in the SDRAM controller conforms to VHDL 87 syntax However, VHDL allows a certain amount of latitude in its construction This latitude provides the engineer with the ability to control the synthesizer's level of optimization of the VHDL code By implementing a greater level of abstraction in the coding methodology, the engineer is able to take full advantage of the optimization algorithms implemented in the synthesizer. Figure 4-7 presents the code shell used to construct the SDRAM controller state-machine.

The code contains two processes, **clock_process** and **state_process** State_process is a purely combinational process that determines the value of the outputs, internal signals, and state of the state-machine Clock_process registers the outputs, state, and internal signals of the state-machine The clock_process guarantees that all outputs will be synchronized as closely as possible with the input clock

The code shell also demonstrates the level of abstraction used in the construction of the SDRAM controller state-machine By using names rather than values for the state notation, the synthesizer is allowed to determine the optimum method of encoding the states. All signals within the state-machine are created with a dual "i" signal As demonstrated in the code shell, only "i" signals are written to in the state_process This practice helps eliminate "race" conditions, and also helps to assure that the synchronous nature of the design will be preserved.

The code shell, when followed, also presents VHDL that is easily readable to those from a more software intensive background.

```
library ieee;

use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

ENTITY entity_name IS
      PORT(CLK     : in  std_logic;
           RESET_F : in  std_logic;
           INPUT   : in  INPUT_0 type;
           OUTPUT  : out OUTPUT_T type);
END entity_name;

ARCHITECTURE behavior OF entity_name IS
      TYPE STATE_TYPE IS (state_0);
      SIGNAL state,         next_state     : STATE_TYPE;
      SIGNAL output_out,    ioutput_out    : OUTPUT_0 type;
      SIGNAL internal_int, iinternal_int : internal_0_int type;
BEGIN

clock_process : PROCESS (RESET_F, CLK)
BEGIN
      IF RESET_F = '0' THEN
            state        <= ;
            output_out   <= ;
            internal_int <= ;
      ELSIF CLK'event AND CLK = '1' THEN
            state        <= next_state;
            output_out   <= ioutput_out;
            internal_int <= iinternal_int;
      END IF;
END PROCESS;

state_process : PROCESS (state, INPUT)
BEGIN
      CASE state IS
            WHEN state_0
                  next_state    <= ;
                  ioutput_out   <= ;
                  iinternal_int <= ;

            WHEN OTHERS
                  next_state    <= ;
                  ioutput_out   <= ;
                  iinternal_int <= ;
      END CASE;
END PROCESS;

OUTPUT <= output_out;

END behavior;
```

Figure 4-7: VHDL State-Machine Code Shell

Chapter 5

Testing and Results

5.1 Software Simulation

The software packages used to develop the software portion of this design (Altera Quartus and MaxPlus II) have the ability to determine the maximum frequency at which the software will function The APEX 20K configuration, consisting of the SDRAM controller, was rated at a maximum clock rate of 59 MHz. The communications interface implemented in the FLEX 10KE device was rated at a maximum clock rate of 58 MHz Therefore, the Memory TIM Module can transfer data at a rate of 116 Mbytes/sec (2 * 58 MHz), which exceeds the required data transfer rate of 75 Mbytes/sec

The software packages can also simulate compiled code, providing the developer with the ability to verify that the program is functioning properly before the PLD is configured Simulations were run for all of the operations described in Section 4 2 2, as well as the transitions between these operations The following timing diagrams illustrate the command output waveforms from these simulations. The RAMCNTL bits are shown for a Row Active and Write (Fig 5-1), Row Active and Read (Fig 5-2), Precharge (Fig 5-3), and Autorefresh (Fig. 5-4) Operation. By comparing these timing diagrams with those supplied by the manufacturer of the SDRAM (see Section 2 3.1 SDRAM Operations), it was verified that the timing characteristics of the SDRAM controller were correct for the commands implemented

5.2 Pulsing-Pin Test

The purpose of the pulsing-pin test was to determine whether the PLDs were operational and configurable For this test, the FLEX10K50E and APEX20K100 were each loaded with a VHDL program that would divide the input clock by 2048, and output that waveform on an easily

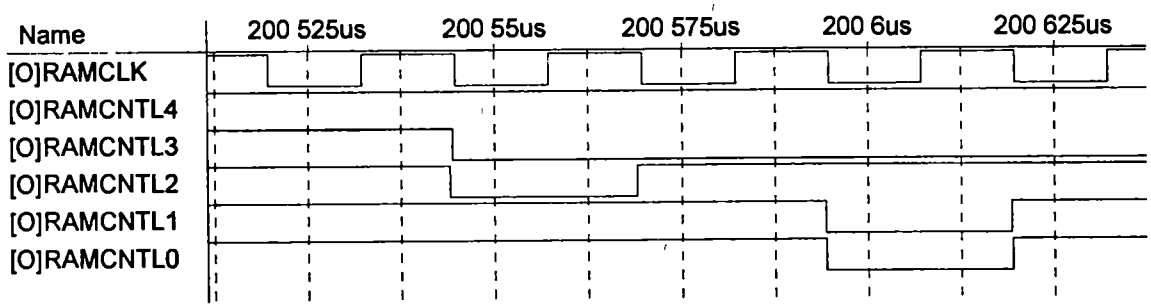

Figure 5-1: Row Active and Write Command Simulation Timing Diagram

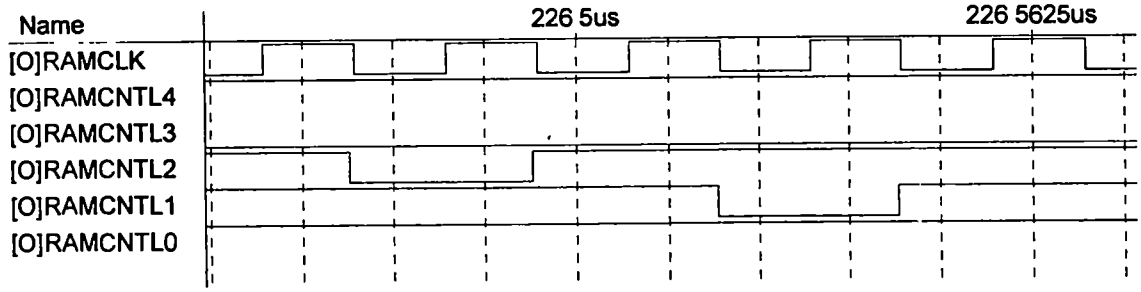

Figure 5-2: Row Active and Read Command Simulation Timing Diagram

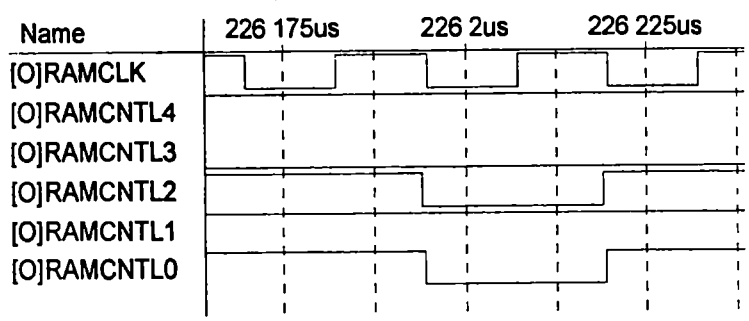

Figure 5-3: Precharge Command Simulation Timing Diagram

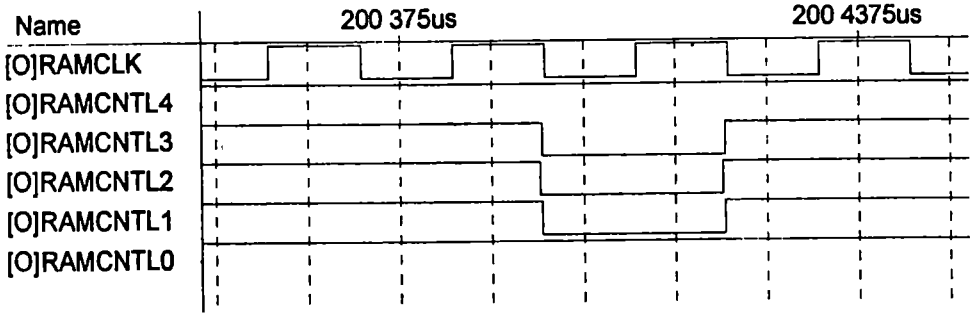

Figure 5-4: Autorefresh Command Simulation Timing Diagram

accessible pin. Both PLDs were configured via the JTAG chain. Upon completion of the configuration, an oscilloscope was used to view the output waveform on the specified pin of each device By this method it was determined that both PLDs were operational and configurable

5.3 Incrementing Data Test

The purpose of the incrementing data-test was to determine whether the SDRAM controller VHDL code was performing properly. The APEX 20K device was configured with the SDRAM controller VHDL code via the JTAG chain The FLEX 10KE device was configured with a modified version of the communications port interface code Instead of reading data from an outside source, the FLEX 10KE would produce incrementing 16 bit words and provide this data to the APEX 20K device. When the memory was full, the contents would be fed from the APEX 20K device to the PC, via the FLEX 10KE device. The data was then checked to verify that data was reaching the PC, and that the data was uncorrupted By this method it was determined that the SDRAM controller VHDL code was performing correctly.

5.4 Sine Wave A/D Input Test

The purpose of the sine wave A/D input test was to determine whether the SDRAM TIM module was capable of receiving data from another TIM module The test also showed whether the SDRAM TIM module was correctly controlling the FIFOs located on the A/D TIM module from which it was receiving data

A function generator was used to feed data to an A/D TIM module. The module converted the input levels to digital values and stored these values in the FIFOs located onboard the module. When the FIFOs denoted that they were almost full, the SDRAM module read 1024 16-bit words from the FIFOs and stored them in the SDRAM After the SDRAM was full, data was read from the SDRAM and written to the PC The output data file was then used to reconstruct the input waveform By comparison of the original input sine-wave to the reconstructed wave-form, it was determined that the SDRAM TIM module was capable of acquiring data from another TIM module, and that it was correctly controlling the FIFOs located on the A/D TIM module Figure 5-1 illustrates the component connections for the test

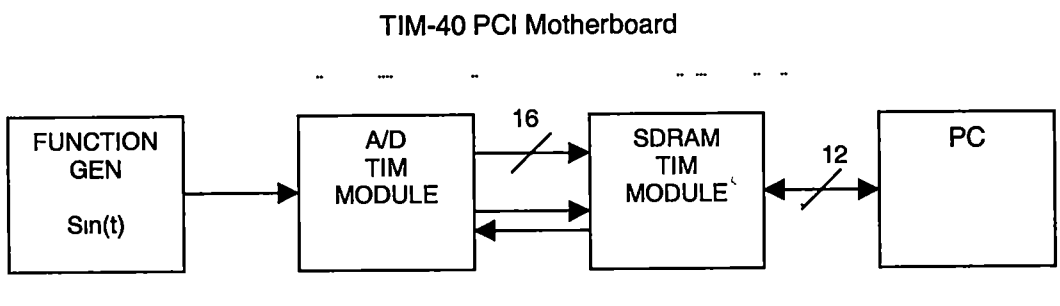

Figure 5-5: Sine Wave A/D Input Test Component Connection Diagram

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

The Memory TIM Module (shown in Figure 6-1) was constructed and its functionality was verified through simulation and testing The module is capable of storing data received from an A/D TIM module at a rate of 116 Mbytes/sec using a custom defined data transfer protocol over two communication ports. Only 5% of the APEX device's total resources were used for the SDRAM controller and 6% of the FLEX device's total resources for the communications interface, making a great deal of resources still available for reconfigurable applications Also, all of the memory elements within both devices are currently unused and available The module conforms to the TIM-40 module specifications, and provides multiple configuration options.

6.2 Recommendations

In the future, it may be desirable to increase the total available storage on the module from 256 to 512 Mbytes The board was designed for the possibility of replacing the 16Mbyte SDRAM chips with pin compatible 32Mbyte chips when they became available on the market. This was accomplished by running a currently unused 13th address line to all the SDRAM chips If the PCB was populated with these 32 Mbyte SDRAM chips, modification of the SDRAM controller VHDL code would be quite simple The user would simply need to implement the use of the expanded memory bus

The current performance of the SDRAM controller VHDL code could also be improved Specific places for optimization might include the auto-refresh and precharge timing, as well as the implementation of additional data lines A replacement of the state-machine with a microprogrammed controller might also be explored [10]

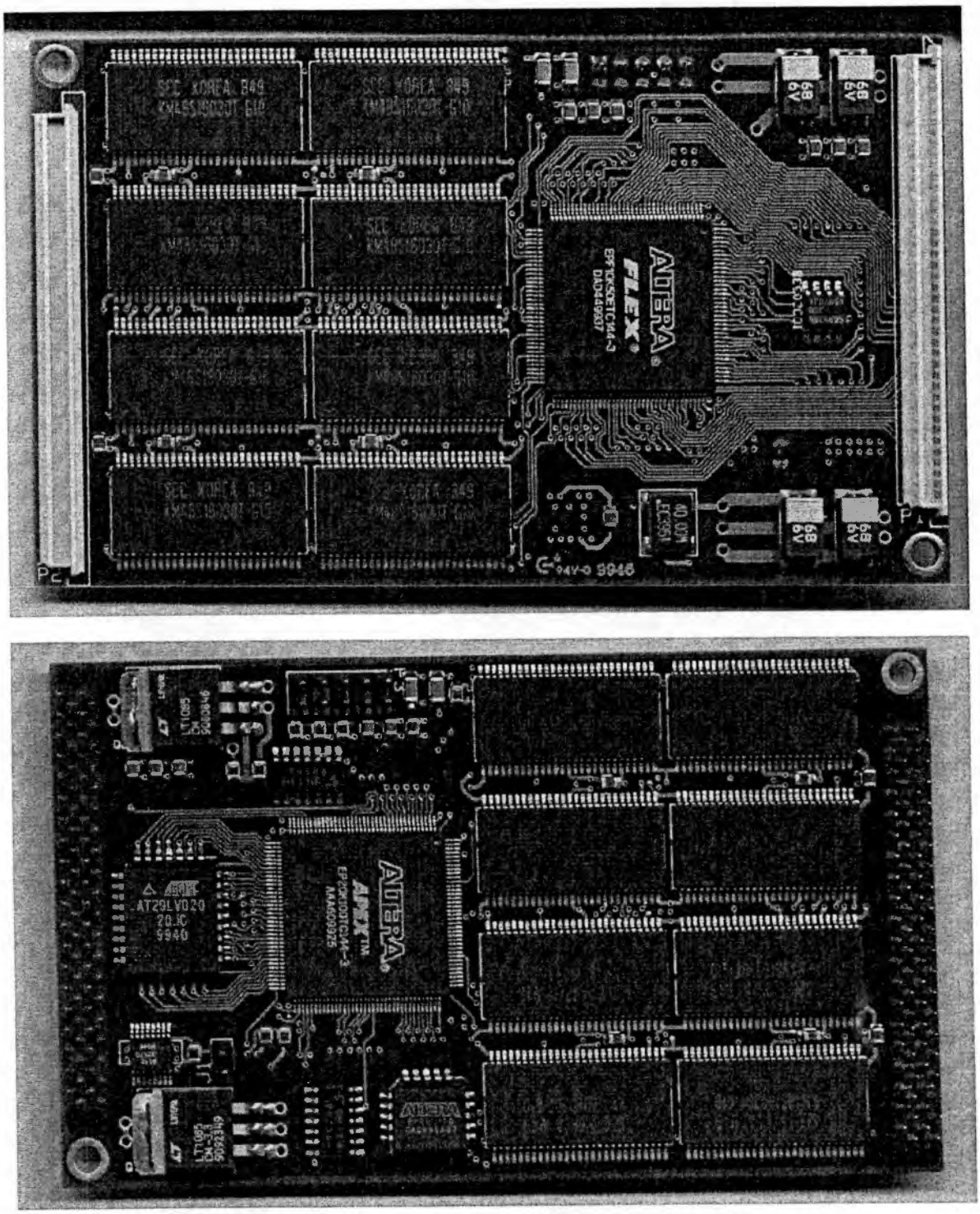

Figure 6-1: Memory TIM Module, Front and Back (not to scale)

Because only 5% of the APEX device's total resources and 6% of the FLEX device's total resources were used for the SDRAM controller, additional functionality could be implemented in the device Such functions might include data modification, error detection and correction, or signal processing

REFERENCES

REFERENCES

[1] "TIM-40. TMX320C4x Module Specification," v1 00, Texas Instruments Corporation, Houston, TX, 1992.

[2] "KM48S16030 CMOS SDRAM, Preliminary," Data Sheet, rev 2, Samsung Electronics, San Jose, CA, March 1998

[3] "Device Operations - V, CMOS SDRAM," Data Sheet, rev 0, Samsung Electronics, San Jose, CA, October 1998

[4] "Sychronous DRAM," Data Sheet, Rev 3/99, Micron Technology, Inc , Boise, ID, March 1999

[5] "Timing Diagram - V, CMOS SDRAM," Data Sheet, rev 0, Samsung Electronics, San Jose, CA, October 1998.

[6] Jerry C Whitaker, The Electronics Handbook, Beaverton, OR Technical Press, Inc., 1996

[7] "APEX 20K Programmable Logic Device Family," Data Sheet, v2 02, Altera Corporation, San Jose, CA, August 1999

[8] "IEEE 1149 1 (JTAG) Boundary Scan Testing in Altera Devices," Application Note 39, v4 02, Altera Corporation, San Jose, CA, November 1998

[9] H. W. Johnson and M Graham, High-Speed Digital Design A Handbook of Black Magic, Upper Saddle River, NJ Prentice Hall, 1993

[10] B.W Bomar, "Implementation of Microprogrammed Control in FPGAs," IEEE Trans Ind Electronics, to appear

VITA

Brannon Paul Wells was born in Grand Forks, North Dakota on March 3, 1976 He completed his first two years of elementary school in Dickinson, North Dakota The rest of his elementary, junior high, and high school career took place in Minot, North Dakota. After completing his high school education, Brannon attended North Dakota State University in Fargo, North Dakota in order to pursue his Bachelor of Science degree in Computer Electrical Engineering Upon graduation from NDSU in December 1998, he decided to avoid working in the "real-world" for at least another year and attend The University of Tennessee Space Institute. Brannon completed his Master of Science Degree in Electrical Engineering in May 2000 He is currently employed with Motorola in Boynton Beach, FL