FAULT TOLERANT FPGA CO-PROCESSING TOOLKIT

DOUGLAS MICHAEL DISABELLO

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

BOSTON UNIVERSITY

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Thesis

FAULT TOLERANT FPGA CO-PROCESSING TOOLKIT

by

DOUGLAS MICHAEL DISABELLO

B.S., Boston University, 2004

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science

2006

Approved by

First Reader _____

Martin Herbordt, Ph.D.

Professor of Electrical and Computer Engineering

Second Reader _____

Allyn Hubbard, Ph.D.

Professor of Electrical and Computer Engineering

Third Reader _____

Wei Qin, Ph.D.

Professor of Electrical and Computer Engineering

Fourth Reader _____

Michael Lovellette, Ph.D.

Research Physicist United States Naval Research Laboratory

The first steps toward widsom are the most strenuous,
because our weak and stubborn souls dread exertion
(without absolute guarantee of reward) and the unfamiliar.
As you progress in your efforts, your resolve is fortified
and self-improvement progressively comes easier. Epictetus

Acknowledgments

I would not have made it with out the support of my friends and family.

FAULT TOLERANT FPGA CO-PROCESSING TOOLKIT

DOUGLAS MICHAEL DISABELLO

ABSTRACT

The most advanced radiation hardened microprocessors are orders of magnitude slower than terrestrial microprocessors. Mitigating the effects of the ionizing radiation found in orbit requires special and often proprietary design techniques by CPU manufacturers causing a large computational performance gap. Commercial Off the Shelf Field Programmable Gate Arrays, for specific compute intensive applications, can fill and exceed this performance gap while retaining fault tolerance. In terrestrial computation, speed-ups of factors of 100x to 1000x are common for appropriate applications; the speed-ups in ionizing radiation environments could be 10 times that. Although FPGAs have previously been used in space to provide particular functions, there is not yet a general capability of space-based FPGA Fault Tolerant co-processing. The Fault Tolerant FPGA Co-processing Toolkit fills this need. Systems containing as few as one FPGA to multiple FPGAs on one board can be utilized with the toolkit to perform co-processing tasks in ionizing radiation environments while providing continuous service.

Contents

List of Figures

List of Abbreviations

ASIC	.....	Application Specific Intergrated Circuit
BRAM	.....	Block Random Access Memory
CLB	.....	Combination Logic Block
FPGA	.....	Field Programmable Gate Array
GRM	.....	General Routing Matrix
HDL	.....	Hardware Description Language
IOB	.....	Input/Output Blocks
ISA Bus	.....	Industry Standard Architecture
JTAG	.....	Joint Test Action Group
LEO	.....	Low Earth Orbit
LET	.....	Linear Energy Threshold
LUT	.....	Look Up Table
NPS	.....	Naval Post Graduate School
NRL	.....	Naval Research Lab
SAA	.....	South Atlantic Anomaly
SEFI	.....	Single Event Functional Interrupt
SEU	.....	Single Event Upset
SRAM	.....	Static Random Access Memory
TMR	.....	Triple Modular Redundancy
VHDL	.....	Very high speed integrated circuit HDL

Chapter 1

Introduction

1.1 The Space Computation Performance Gap

Computation in space presents well-known problems. The most advanced radiation hardened microprocessors are orders of magnitude slower than terrestrial microprocessors. Mitigating the effects of the ionizing radiation found in orbit requires special and often proprietary design techniques by CPU manufacturers causing a large computational performance gap. For example, the most advanced CPU from BAE Systems is currently a 133 MHz RISC processor; an increase only to 250 MHz is projected in the near future. As a result, the bare minimum of computation is performed in space, with large amounts of data being transferred to the ground for processing. These large data transfers from orbit are slow compared to terrestrial communications and are further compounded by limited terrestrial receiving stations on the Earth's Surface which results in a limited number of opportunities of finite time period data transfer windows.

Commercial off-the-shelf Field Programmable Gate Arrays, for specific compute intensive applications, can fill and exceed this performance gap while retaining fault tolerance. In terrestrial computation, speed-ups of factors of 100x to 1000x are common for appropriate applications; the speed-ups in ionizing radiation environments could be 10 times that. Having this high performance computation available in space will allow for better use of limited communication time. The end result will be the ability to transfer more meaningful data and results during the communication windows-of-opportunity. Other benefits include faster and cheaper development cycles as printed circuit boards containing FPGAs are only limited in functionality by the designs instantiated. FPGAs also offer the unique

ability to change function in flight to accommodate changing mission requirements.

Although FPGAs have previously been used in space to provide particular functions, there is not yet a general capability of space-based FPGA Fault Tolerant co-processing. The Fault Tolerant FPGA Co-processing Toolkit fills this need. Systems containing as few as one FPGA to multiple FPGAs on one board can be utilized with the toolkit to perform co-processing tasks in ionizing radiation environments while not requiring the design of entirely new FPGA designs and software. The toolkit was created in modular form allowing and facilitating the addition of new components for increasing the breadth of available functions. End users will be able to spend development time implementing co-processing algorithms and not the underlying fault mitigation support structure.

Chapter 2

Background

2.1 Field Programmable Gate Array Architecture

To quote Tom Van Court,"FPGAs are a giant bag of logic components" from which the FPGA designer connects and manipulates the components to give desired logic circuit functionality. FPGAs offer short development cycles and can be programmed an unlimited number of times. For specific applications FPGAs offer large speeds up over current terrestrial PCs by exploiting inherent fine grained parallelism. Computational intensive tasks have gained 200 to 1600-fold speed ups. [1] [2]

Xilinx is the leading configurable logic manufacturer. Xilinx FPGA architecture is comprised of configuration logic blocks (CLBs), input/output blocks (IOBs), and other specialized components. CLBs "provide the functional elements for constructing logic" functions. [3] CLBs are comprised of programmable Look Up Tables (LUTs) and supporting logic to give desired functions. IOBs "provide an interface between the package pins and the CLBs". [3] Specialized components include dedicated Block RAMs (BRAMS) that provide on chip memory and tri-state buffers. These different architectural components are all interconnected by the general routing matrix (GRM). The GRM uses arrays of routing switches to achieve this functionality.

FPGAs use SRAM cells to create a design instantiated into the device for a given functionality. These cells control the function of the CLBs, IOBS, and the general routing structure between these elements and other elements. They are referred to as the configuration memory of an FPGA. The configuration memory is programmed on device power up and can be reprogrammed to load new designs into the FPGA.

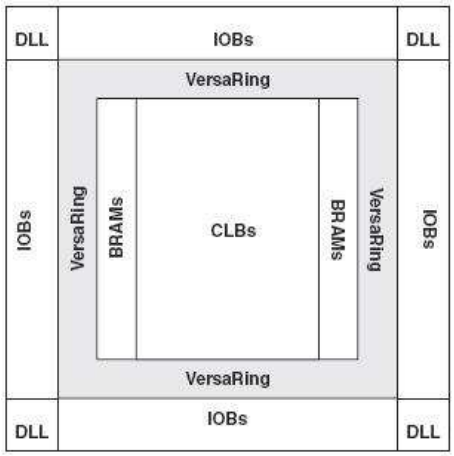

Figure 2·1: Xilinx Virtex FPGA Architecture [3]

FPGA designs are normally described in Hardware Description Languages. The two most common languages are VHDL and Verilog. Synthesis, place and route, and BitStream generation tools interpret the HDL source code and ultimately create configuration data streams to perform the same function as the HDL description in the FPGA. A configuration data stream design is stored in a BitStream file that is programmed into a FPGA using the SelectMap or JTAG interface and connects to the FPGAs internal configuration Finite State Machine controller. The SelectMap interface allows 8 bit wide parallel configuration data loading while the JTAG interface allows serial configuration data loading. The configuration controller interprets commands in the BitStream and stores the BitStream configuration data to the configuration memory.

2.2 Ionizing Radiation

Satellites are continually exposed to particles associated with ionizing radiation. Hardware payloads are susceptible to faults induced from interactions with these particles. The first documented observations of these faults occurred as far back as 1975. [4] The susceptibility of these systems to ionization radiation only grows as transistor features sizes and threshold voltages continue to decrease following Moore's Law.

Low Earth Orbit, roughly defined as the range from 500km to 2000km above the Earth's

surface, contains ionization particles comprised of greater than 93 percent protons, approximately 6 percent alpha particles, and one percent heavy nuclei. [4] The three main sources of particles include trapped radiation belt particles, cosmic rays, and solar flares. [5] The Earth's magnetic field is responsible for trapping these particles within these radiation belts referred to as Van Allen belts. The particles spiral back and forth traveling along the magnetic field lines. Due to the off center dipolar field of the Earth, the Van Allen belts reach their lowest altitudes in what is referred to as the South Atlantic Anomaly. Mission systems incur a disproportionately high level of faults during this portion of their orbit due to the much higher density of particles located at lower altitudes [5].

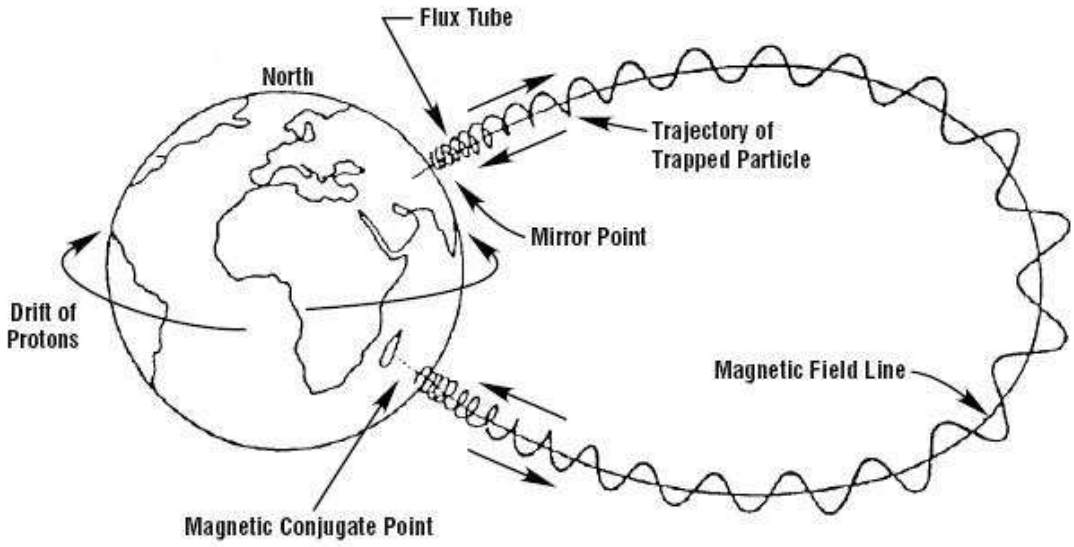

Figure 2·2: Particles spiral back and forth across magnetic field lines [5]

Ionizing radiation particles cause faults when they strike the semiconductor substrate of transistors. These particles can produce direct and secondary nuclear reactions resulting in generation of excess charge in the substrate. These extra free electrons and holes can be picked up by the drain diodes of transistors; when the energy level of the interacting particle is high enough it will cause a logic state change in a circuit node. [4] The minimum amount of energy needed to cause a logic state change is referred to as the Linear Energy Transfer Threshold. The Linear Energy Transfer refers to the rate at which a particle interacting

with the substrate loses its energy. This is also referred to as stopping power. [4] [6]. As feature sizes on chips continue to decrease the probability of a particle striking an area of the substrate and interacting with a transistor increases. This increase occurs because of the more densely populated layout leaving less area of the substrate where particle interactions would strike an"empty" part of the silicon substrate. Decreased threshold voltages also result in increased sensitivity to ionizing particles. Interactions in a chip with a larger threshold voltage would result in some lower energy particles not adding enough energy to leave a circuit's noise margin and effectively having the particle's upset be ignored as ambient noise. A smaller threshold voltage would cause that same particle to now add enough energy to leave the noise margin of a circuit and result in a change of logic state. A change in the logic state of a circuit due to ionizing radiation is referred to as soft error when no permanent damage is done to the circuit. These soft errors are commonly referred to as Single Event Upsets (SEUs). A Single Event Functional Interrupt (SEFI) is defined as a SEU that results in a system no longer being able to function. Hard errors encompass interactions that result in permanent damage to the chip. Single Event Hard Errors include latchup, burnout, and transistor gate rupture. A device's single event error response is characterized by its cross section. Cross section is measured by number of errors per ion fluence and given in cm^2 per bit or device. [6] Increases in SEU rates, permanent damage, and complete failure to a circuit are known to occur when Total Ionizing Dosage reaches a certain threshold for a particular chip.

2.3 Ionizing Radiation Field Programmable Gate Array Effects

Xilinx Field Programmable Gate Arrays are primarily based on the same technology used in static random access memory (SRAM) and therefore are sometimes referred to as SRAM FPGAs. To better understand particle interactions causing the change of a stored logical state at the transistor level one can compare FPGAs to a standard SRAM cell. The standard SRAM two inverter feedback loop uses regenerative feedback to hold a logic state. If a particle interaction causes a node to transition it may propagate through to the

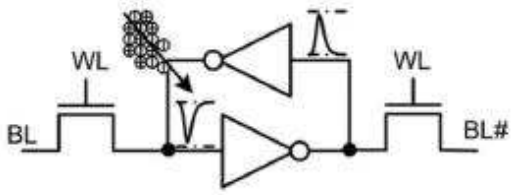

Figure 2·3: Upset SRAM Cell [4]

first inverter to the second node and the second inverter driving the first node towards an opposite logical value. Eventually this regenerative action will cause both nodes to flip and an inversion of the value stored in the memory cell. [4]

From a broader perspective FPGAs are susceptible to not only SEUs occurring to areas such as stored register values and combinational logic, but also the configuration memory and programmable interconnect points that store and form the instantiated design. The majority contribution for FPGA SEUs comes from the configuration memory and accounts for 91 percent of a typical FPGA cross section [7]. Of the configuration errors that cause design failures 78 - 84.8 percent were caused by upsets in the routing structure and the remaining 20 percent were caused by upsets in the control bits and CLB LUT value changes. [4] Changes to portions of the design (such as values in a look up table) can cause errors that change the intended function of a design while still allowing for continued design operation. For a simple example, errors in a look up table could result in a CLB operating as a NAND instead of a NOR.

There are Single Event Functional Interrupts that occur independent of the instantiated design. When SEUs occur in specific areas of the chip they most often result in large changes to the configuration memory or complete functional interruption of the FPGA (not just the instantiated design). The three areas include the JTAG TAP controller, configuration controller state machine reset, and the Selectmap configuration pin upsets. [4] The total cross section for these three possibilities is measured at less than 1^{-5} cm^2 and require an LET threshold between 8 to 16 MeV-cm^2/mg. [4][5] The JTAG TAP

contains a FSM and logic used for component testing and programming that upon upset can cause an FPGA to perform incorrectly or not at all. This is a problem that has also occurred in previous systems and caused SEFIs even in ASICs. However, for most ASICs a pin can be grounded to stop the JTAG FSM from functioning. This is not an available option for Xilinx FPGAs. An upset to the configuration control state machine can result in the configuration memory being cleared. This is a result of the FSM reacting as if the PROGRAM pin was pulsed in which the normal response is to clear the configuration memory. The Selectmap interface used for configuration and readback of the FPGA with an 8 bit wide data path and associated control pins can also be upset. In Xilinx FPGAs the Selectmap interface utilizes dual-purpose pins. If a design configuration does not contain a "persist" option these configuration pins will be turned into normal input/output pins for use by an instantiated design. An upset in the configuration memory can effectively turn all or a portion of the Selectmap pins into general purpose input/output. The ability to configure and read back configuration data using the Selectmap will be lost until the FPGA is completely powercycled or configured using another interface.

Permanent damage and errors can occur in FPGAs. Once a FPGA meets and exceeds its TID a part will continue to develop more errors and eventually cease to function. Latch up errors and contention from SEUs are of much less risk of causing permanent damage for a FPGA. Virtex designs limit the drive current for internal bits to stop contention from causing high current damage. [5] However, the benefit of being able to reprogram a FPGA allows for a design to be place and routed to avoid using the damaged portions of a chip in certain cases.

2.4 FPGA Fault Mitigation Techniques

Faults that occur in FPGAs can be grouped into two main categories. One type is unique to FPGAs and is described as architectural upsets. These upsets cause literal changes to the architecture of the instantiated design. Dynamic data upsets describe the remainder of the possible upsets. These occur when stored values on the chip change during its operation

and include single event transients that become latched in the circuit and become SEUs. Using various fault mitigation techniques described in this section it is possible to achieve a high reliability rate for continuous operation. Without fault mitigation techniques any SEU would in most cases result in a Single Event Functional Interrupt or at the very least incorrect computational results.

Architectural upsets occur as a result of FPGA configuration memory upsets. As previously mentioned, these upsets may not cause a functional interrupt but can cause undesired circuit operation. Although a single upset to certain portions of the FPGA could immediately cause a SEFI, the accumulation of configuration errors over time will most definitely results in loss of service. Correction of these errors before accumulation is a necessity for continuous circuit operation. Xilinx FPGA devices allow for the possibility to correct these errors while the FPGA and the design is still providing service. In normal FPGA operation and programming the active portions of the chip are shutdown, a new configuration data stream is loaded into the configuration memory, and the chip is brought back into service. Xilinx FPGAs allow for what Xilinx refers to as active partial configuration. [8] Partial reconfiguration employs configuration files specially created to only reconfigure portions of the FPGA while keeping it in service. Utilizing this Xilinx feature one can continuously refresh the design in configuration memory repairing architectural upsets while the device is still in service. This fault mitigation technique is most commonly referred to as "scrubbing". [8] In order to institute this technique an external device must configure the initial configuration and possibly utilized for monitoring and correcting upsets. The device would require the ability to program and in some implementations read back the configuration of the FPGA utilizing the selectmap or the JTAG interface. Scrubbing is the continuous rewriting of the configuration memory at a rate faster than the expected SEU rate for a given orbit.

The instantiated design must contain certain design rules in order for scrubbing to be effective while a FPGA is in service. Instantiated designs can not contain distributed RAM or Xilinx primitive SLR shift register modules. These modules use Look Up Tables

in the CLBs to hold dynamic data that changes as the instantiated design operates which is normally static. Scrubbing a design that contains these modules would result in incorrect operation of a design by changing the data in the LUT tables back to initial states while the FPGA is active.

Although scrubbing will repair any architectural design upset that occurs a mechanism is necessary for providing continued service in between the occurrence of an upset and its correction. Triple Modular Redundancy (TMR) can fulfill this requirement. TMR utilizes replication and area to mitigate architectural errors in a design. [9] A module inside a design is created in triplicate and the output of the three identical modules is chosen using a majority voter. SEUs in the configuration memory will occur in one module, but the two unaffected modules will "win" the majority vote and the correct output will result.

TMR and scrubbing can mitigate errors that occur on chip. If possible simple triplication of data onto several pins is the easiest method for off chip communication fault mitigation. However, hardware constraints do not always allow for three pins to triplicate the same data and triplicates of some devices may not be desirable or possible. SEUs can result in FPGAs IOBs changing the intended data direction of a pin or just simply outputting the incorrect logic state.

TMR and scrubbing combined can take the SEU error rate for an FPGA to essentially zero. However, the possibility for SEFIs will still exist in the portions of the FPGA dedicated to configuration. [9] The cross section for these SEFIs is minimal and discussed in the previous section.

Chapter 3

Hardware Development Board

3.1 Description

The United States Naval Postgraduate School created the original development printed circuit board design. The board and its derivates were originally created for a proof of concept Configurable Fault Tolerant Processor for radiation space environments. However, the design was created with well thought out foresight into future applications. It is this foresight that allowed its use in the conception and development of the Fault Tolerant FPGA Co-Processing Toolkit. The PCB used for development was a derivative of the NPS board created by the United States Naval Research Laboratory. The original NPS design utilized two Xilinx Virtex RADHARD FPGAs. These Xilinx FPGAs feature a higher Total Ionizing Dose. The NRL design replaced the second FPGA with a non-RADHARD Virtex 2 FPGA offering a much higher logic density. There is no functional difference between the radiation hardened and commercial versions of the chip. Limited availability and magnitudes of difference in cost were the factors in utilizing a commercial Virtex 2 FPGA. The NRL PCB was manufactured by Silver Engineering and interfaces with an x86 embedded system. As can be seen in the functional diagram shown in figure 3.1, the Virtex FPGA contains physical connections to most components on the board while the Virtex II FPGA primarily contains connections to the Virtex FPGA.

The TMZ104 x86 embedded processor and system board is a product of Tri-M Engineering. A Slackware Distribution of the Linux operating system is installed on the system. The PC104 Bus, which is a derivative of the ISA standard, is the only connection between the NPS PCB and the TMZ104 system. The embedded processor system is used to send

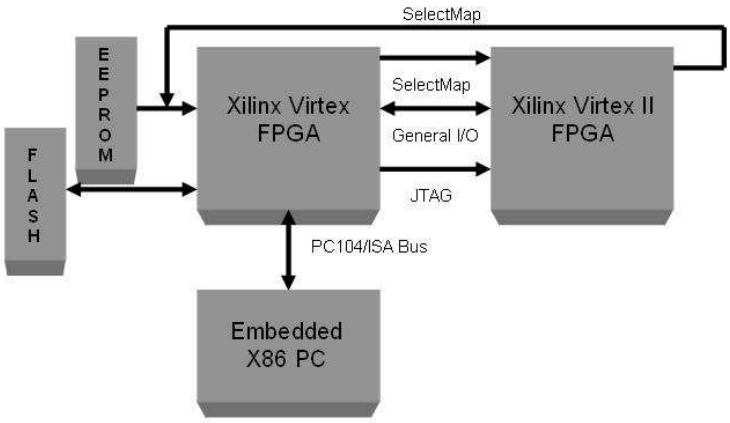

Figure 3·1: NRL Board Block Diagram

and receive data and commands from the NPS PCB. The Virtex FPGA contains the only physical connections to the PC104 Bus.

The NPS PCB has a plethora of configuration options for the FPGAs. Xilinx EEPROMs are located on the board that can be loaded with bitstreams used to initially configure the FPGAs on power up. JTAG interfaces are available to the FPGAs and EEPROMs for configuration and debugging applications. Both FPGAs also have physical connections to each other's Selectmap and JTAG pins giving the possibility of one FPGA to configure another.

The only onboard memory outside of the FPGAs is an Intel 32Mb Flash with physical connections to both FPGAs. The data stored on Flash chips is impervious to Single Event Upsets, however the controller embedded in the device is not. Experiments have shown upsets in the controller cause incorrect device operation and in some instances permanent device failure. [10] Communication between the FPGAs is accomplished utilizing 43 general I/O pins. Both FPGAs share physical connections to a 50 MHz oscillator, which removes any need to be concerned with transferring data between different clock domains.

3.2 Radiation Testing

The Naval Postgraduate School and Naval Research Lab developed a testing strategy to observe radiation effects and prove fault mitigation techniques utilizing the development boards previously described. These tests were conducted at the Crocker Nuclear Laboratory at University of California, Davis. A cyclotron was used to emit protons and the test boards were exposed to the resulting beam. The setup was chosen to emulate the effects the FPGAs will see in actual orbit.

Test designs were created to monitor and record the number of SEUs that occurred. Designs to meet this goal included shift registers to utilize all the FPGA resources and then report upsets in real time. Other designs tested scrubbing techniques and TMR as fault mitigation techniques. A limited variation of a pix processor using TMR was included in this grouping of tests. The results of the tests proved that scrubbing and TMR is viable way to repair upsets while providing uninterrupted service. SEUs were corrected while the FPGAs were in service and under radiation radiation exposure. Correct data outputs were also observed while SEUs were occurring in the test designs.

The two boards tested included one with two Virtex Radiation Tolerant FPGAs and the NRL derivative with one Virtex II FPGA in place of one of the Virtex's. Exposure was limited to one FPGA per board and allowed for radiation testing of both the Virtex (XQRV600) and the Virtex II (X2CV6000). A final exposure setting of 8.42 x 10^5 protons/cm^2-sec was found to cause an SEU every 10 - 30 seconds for the Virtex FPGA. The Virtex II had an SEU occur every 1.69 seconds for the same fluence. [11] The difference in SEU occurrence is believed to be the result of the smaller feature size of the Virtex II.

Flight versions of the test beds will be flown on two different satellites. One of the satellites is referred to as NPSAT1 and will orbit at 560 km, 35.4 degrees inclination.[11] This orbit causes the NPSAT1 to travel through the South Atlantic Anomaly (SAA) 15 minutes out of every 96 minute orbit. The correlation between upsets and fluency gathered at the Crocker Nuclear Laboratory can be thought of as linearly proportional to expected

upset rates in orbit. Therefore, the expected upset rate for the given orbit based on time in the SAA is approximately 1 upset every 5 days for the Virtex and an order of magnitude greater for Virtex II. [11]

Chapter 4

Fault Tolerant FPGA Co-Processing Toolkit

The FPGA Based Coprocessor Fault Mitigation Toolkit was researched and developed to meet three goals. The first goal was to create a support system to be instantiated into an FPGA using a hardware description language (HDL) and the creation of software to enable an embedded host PC to communicate with the FPGA coprocessor system. This support system was designed to manage all underlying tasks leaving an end user to only develop and manage the HDL module(s) responsible for a given coprocessing algorithm. These functions include the transferring of data to and from a Host PC and all FPGAs, initial configuration and scrubbing of all FPGAs, and interfacing with any secondary physical board components. The support system was created utilizing modular HDL interfaces for different physical components and resources available on the previously described Naval Research Lab development board. This modular design allows for easier integration of new HDL modules for unknown future enhancements to next generations of coprocessing boards.

The second goal was to develop a fault tolerant version of the support system previously described. The result will contain all the functionality of the previous system while now allowing uninterrupted service in SEU inducing radiation environments. This required changes to the HDL modules for the FPGAs and the software on the host PC. The fault mitigation techniques utilized in creating the fault tolerant version of the support system were then used to facilitate reaching the last goal. The third goal was to give an end user not only a fault tolerant support system, but facilitate the migration of previously created coprocessing HDL designs to fault tolerant versions and the creation of new fault tolerant coprocessing HDL designs. This facilitation is accomplished by giving HDL templates

and completed components, guidelines for new HDL component creation, and Host PC programs and C++ header files to the end user. The end user can then use the tools to not only create new HDL coprocessing designs, but also new HDL support components for next generation coprocessing boards as new physical resources are added.

The net result of these three goals was the development of HDL components and software programs to facilitate physical platform independent fault tolerant co-processing. Systems containing as few as one FPGA to multiple FPGAs on one board can be utilized with the toolkit to perform coprocessing tasks in SEU inducing radiation environments while not requiring the design of entirely new HDL components and software. The toolkit was created in modular form allowing and facilitating the addition of new components for new functionality. End users will be able to spend their development time implementing co-processing algorithms and not the underlying fault mitigation support structure.

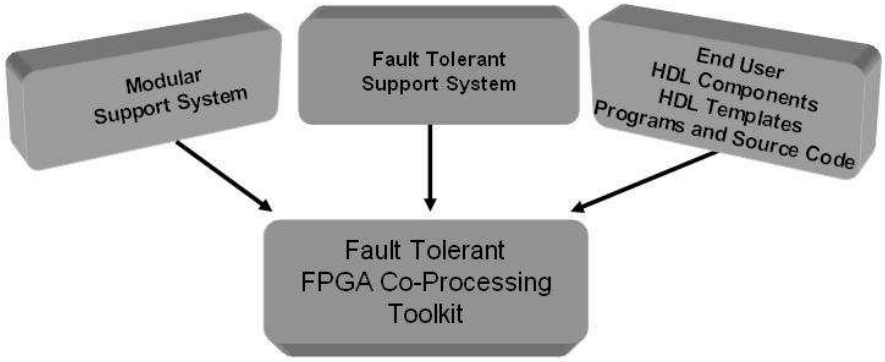

Figure 4·1: Toolkit Intermediate Design Goals

Chapter 5

Support System

The support system is a collection of HDL components and host PC software developed on the Naval Research Lab board described in the test bed description section. Its purpose is to integrate all the physical components on the board for use in fault tolerant co-processing support functions. This includes not only data transfers between these components, but also the initial configuration and scrubbing of the Xilinx FPGAs. The physical components include the Virtex FPGA, Virtex II FPGA, Flash Memory, PC104/ISA Bus, and the x86 host PC. The HDL components allow for interfacing to and controlling these physical components. The NRL board design has all physical interconnects to components centralized around the Virtex FPGA. This FPGA is therefore utilized as the "support FPGA" containing all the HDL modules for data routing and control. The support FPGA is programmed using the JTAG interface and an external PC or a EEPROM can be programmed with the initial configuration that would program the support FPGA on power up. The Virtex II FPGA is utilized as the coprocessing FPGA and would contain the HDL coprocessing modules. The coprocessing FPGA is programmed using the support FPGA.

Creation of the HDL components for instantiation in the FPGAs was accomplished using VHDL, Xilinx Integrated Software Environment (ISE), ModelSim, and Chipscope Pro. VHDL or Very high speed integrated circuit Hardware Description Language is used to describe the support components that are instantiated into the FPGAs. Xilinx ISE compiles, synthesizes, and place and routes the VHDL designs to the FPGA fabric for the targeted FPGA architecture. ModelSim is a software product of the Mentor Graphics Corporation used in the functional simulation of VHDL components. Xilinx's Chipscope Pro inserts cores into FPGAs along side instantiated designs to allow real time logic analysis.

Chipscope is used to debug errors that can not be detected in the simulation environment.

The x86 embedded host system has a Slackware Linux distribution operating system installed. Therefore, the software for the support system was also developed on x86 distributions of the Linux Operating System. The C++ programming language described the programs compiled using the GNU GCC project compiler.

5.1 VHDL Components

In figure 5.1 the full HDL support system and interconnects is visualized. The following subsections describe the individual VHDL components for the support system and their functions.

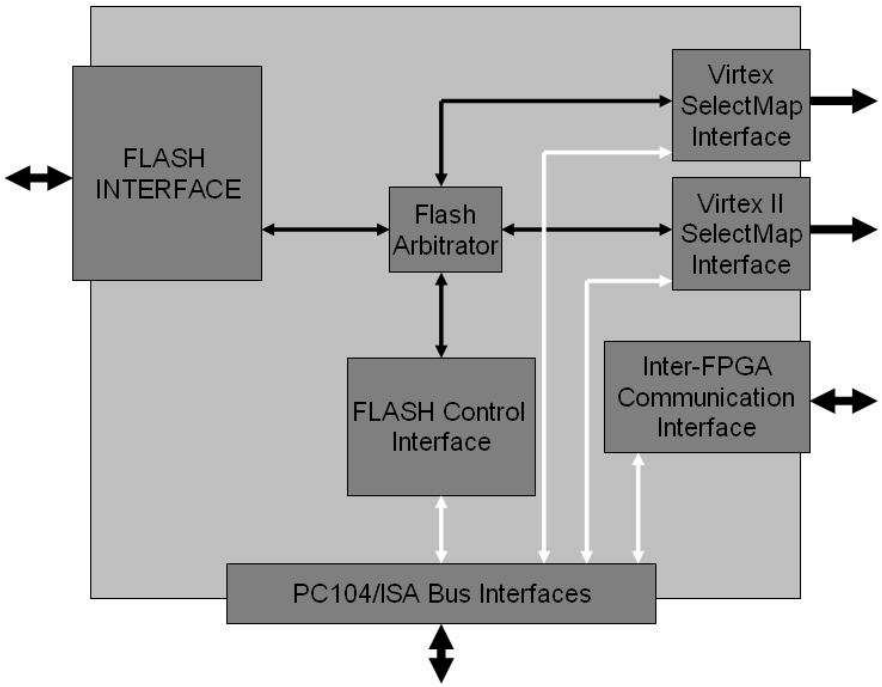

Figure 5·1: Support System Functional Schematic

5.1.1 PC104 / ISA Bus Interface

The only connection between the embedded host PC and co-processing board is the PC104 Bus. The PC104 Bus is a based directly on the ISA Bus standard. The PC104/ISA Bus interface is responsible for all data transfers on and off the FPGA to the embedded host PC. The bus allows for 8 bit data transfers synchronized by the WRITE and READ signals. The AEN signal denotes I/O transfers vs. memory transfers. The interface monitors for logic state transitions on the write signal and then samples the data pins during the period where the write signal is set to logic '0'. The same is done in reverse for reads from the interface except now the data pins are driven by the interface. A Bus clock, extended data transfer, and DMA signals are also available. However, these were not needed for the current implementation.

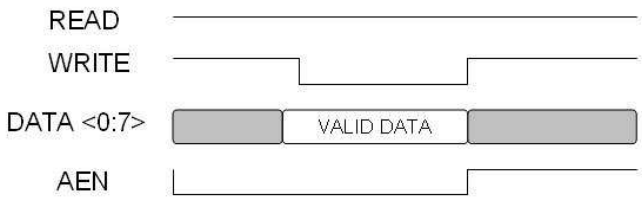

Figure 5·2: PC104/ISA Bus Write Data Transaction Waveform

The interface uses two FIFOs created around Xilinx instantiated BRAMs to buffer all in coming and outgoing data transactions. The FIFOs allow for simultaneous read and writes, have empty and full flag signals, and ignore new data when full. Data output from the FIFO is available the next clock cycle after a read request.

The external portion of the bus interface requires two addresses from the host PC addresses space. The data address is used to send and receive data from the interface. The second address is referred to as the control address. A write to the control address sends a synchronous reset signal to the interface and also causes an output reset port to go high. The output reset port can be utilized by other interfaces for initial resets. A read from the configuration address returns the current state of the empty and full flags for each FIFO to

the host PC. This is used to monitor when new data is available on the bus and to prevent overflows of the FIFOs.

The internal side of the interface has module ports for write, read, data in, data out, and status flags. The status flags are combinational and display the current state of both the input and output FIFOs. Modules that connect to a bus interface monitor the flags to wait for new data to arrive or for the output FIFO to be read out by the host PC before continuing to write to it.

Each of the main VHDL components described in this section use a copy of this interface to obtain its own address space on the bus. This allows for easy addition and/or removal of components to the system. It also allows for simplified access to individual components by giving the host PC direct access to individual components through different addresses.

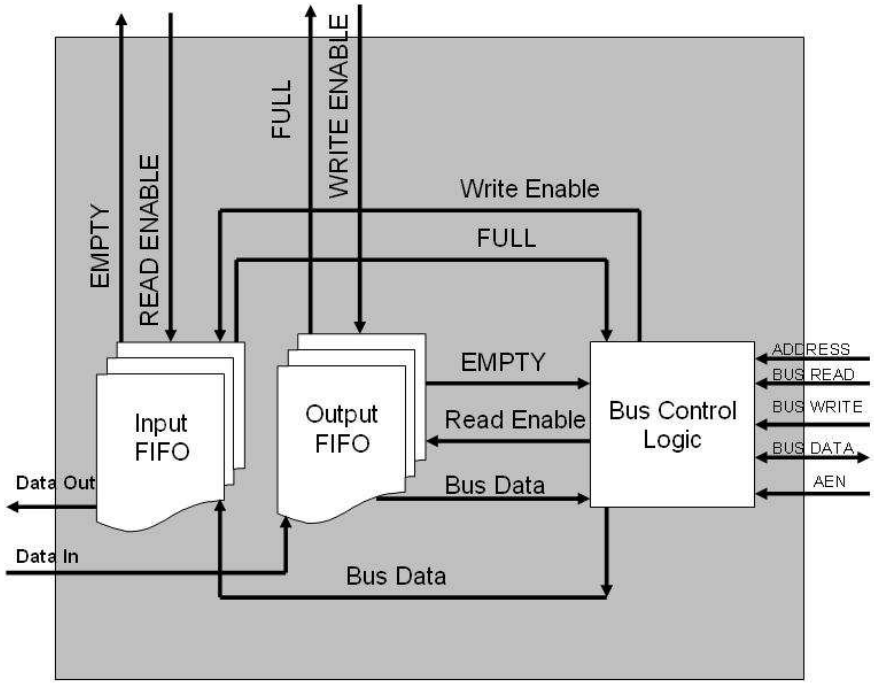

Figure 5·3: PC104/ISA Bus Interface Schematic

5.1.2 Flash Interface

The flash interface accepts commands and data from other VHDL modules and translates them into a series of signals to perform the desired functions in the flash memory. Flash memories do not have an external clock. All signal changes and data exchanges occur with predefined timing constraints and are preceded by commands to setup certain operations. [12] Commands and timing constraints are used in writing and reading from the flash chip. The data storage in flash is partitioned into blocks. Flash memories are erased by clearing

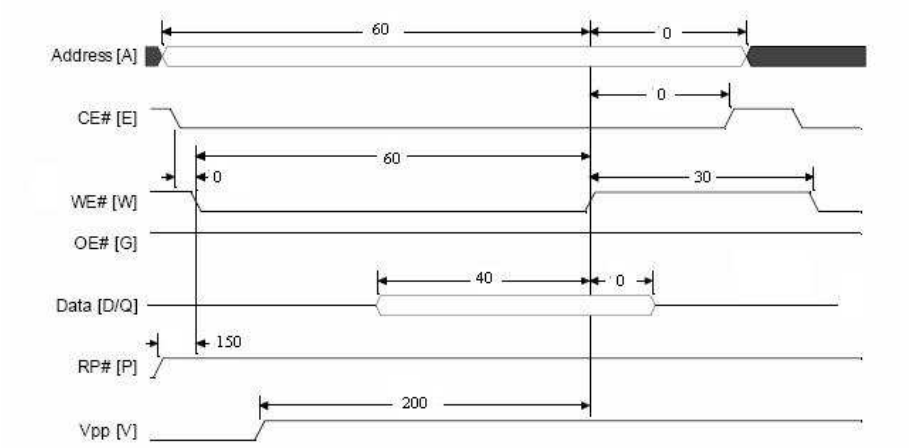

Figure 5·4: Flash Timing Schematic (ns)[12]

individual blocks of storage. When a block is cleared all the bits in the block are reset to logic value of '1'. Programming a block changes logic ones to logic zeros as necessary to store incoming data. Each individual block in the flash memory can also be locked and unlocked. Locking a block prevents any writes and the block can not be erased.

The flash interface has five valid commands: read, write, lock block, unlock block, and erase block. When a read command is registered the flash interface will fetch the data at the requested address and will then display the data on the internal port and raise a synchronous data valid flag signal. Modules that use this function issue the command and then sample the returned data when the valid signal gives a logical '1' value. The block lock, unlock, and erase commands can only be executed when the flash address is set to the

beginning of a storage block. For the Intel Flash on the NRL board, flash block beginning addresses are multiples of 0x7999. At the end of the memory, the last few blocks are known as 'boot blocks' and have a smaller size. All commands when executed by the interface raise an interface busy flag signal. This prevents modules from issuing new commands that would otherwise be ignored by the interface if still busy executing the previous command.

5.1.3 Flash Interface Arbitrator

The Flash control and Selectmap interfaces, described in following sections, all require access to the flash. The Flash interface arbitrator assigns priority levels to each of the modules and gives access permission to the flash interface command bus based on the priority levels. A module gains access by requesting it and waiting for the arbitrator to output its priority identification number. Once a module gains access it will retain access until it deasserts its request signal. This prevents changes to data while the flash is being accessed by the Selectmap interfaces. Changes to the flash data while a Selectmap interface is actively programming a FPGA would cause the programming to fail.

5.1.4 Flash Control

The Flash Control module facilitates the interaction between the flash interface module and a PC104/ISA bus module. Commands from the embedded host PC are interpreted and then executed by the Flash interface. Data is routed to and from the host PC to the flash interface and ultimately the Flash memory. The Flash control takes 8 bit data transfers from the ISA bus interface and reorganizes them into 16-bit data words for use with the Flash interface module. The following is a list of commands that the host PC can issue and their function.

- **Load Address:** The host PC sends 0x07 followed by three bytes. The three bytes contain the flash address to be loaded into the Flash control module. The control module only uses the first 5 bits of the third byte. The Intel flash address is 21 bits long.

- **Increment Address:** The host PC sends 0x08 and control module increments the flash address stored in the control module.

- **Read:** The read command, 0x00, uses the stored address in the flash control module to retrieve the data from the flash memory through the flash interface. The control module then sends the retrieved data to the output FIFO of the bus interface for retrieval by the host PC.

- **Write:** The write command, 0x01, is followed by 4 additional bytes. The 4 additional bytes comprise a 32-bit word equal to the number of 16-bit flash words that will be transferred across the bus and written to flash. The host pc then transfers all the data in a single burst across the bus. By setting the 32-bit value to zero a single flash word can be programmed. During burst writes the address in the control module is automatically incremented. It is important to note that before a large burst command across several blocks can be accomplished the blocks must have been previously unlocked and erased.

- **Lock, Unlock, and Erase:** These commands simply mirror the commands available in the Flash interface module. They are also only valid when the address is set to a beginning flash block address.

- **Flash Address to Bus:** This command, 0x09, sends the current flash address stored in the control module to the output FIFO of an ISA bus interface. This command is mainly used for debugging purposes.

5.1.5 Selectmap Interface Modules

The Selectmap interface modules control configuration data flow to the FPGAs configuration interfaces and are capable of initial and repeated configurations with out power cycling the FPGA. The Selectmap modules merely control signals to the Selectmap interface and repeat configuration data from another source. The configuration data contains commands to the configuration FPGA controller and the actual configuration data. The specifics on

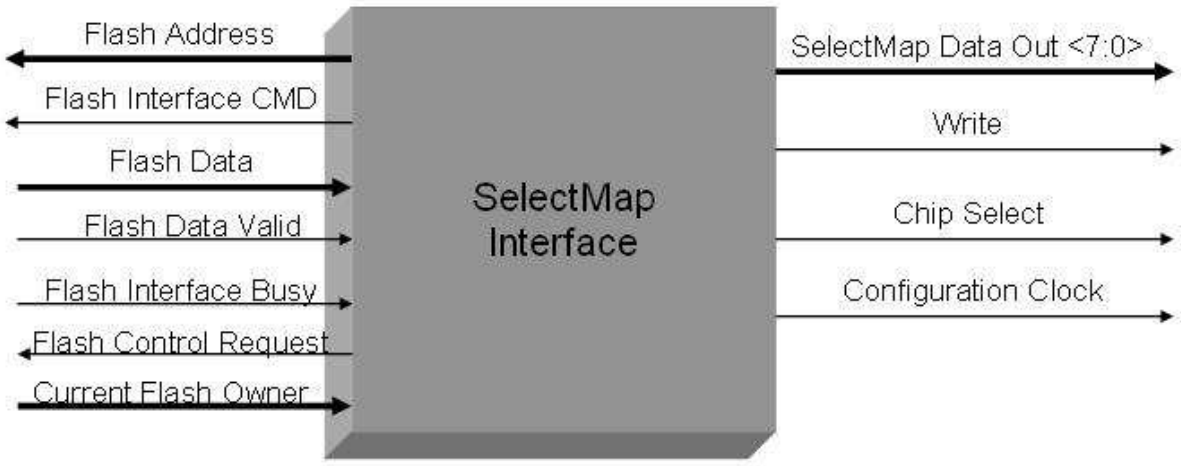

Figure 5·5: SelectMap Interface Schematic

these configuration data stream commands and how they facilitate initial configuration, repeated configurations, and repairing of SEUs to configuration memory is covered in detail under the fault tolerant support system section.

The configuration clock signal is used to control data flow to the FPGA configuration controller. The clock signal is used to register individual values and only transitions after new data has been latched to the Selectmap data pins. This allows non-uniform delays while awaiting configuration data arrival from different sources. Xilinx FPGAs require handshaking above 50 MHz and utilize a busy signal so the configuration interface can process the incoming data. The modules have a clock speed below 50 MHz and do not require any handshaking.

There are two versions of the Selectmap interface. One version works with the first generation of Virtex FPGAs and the second works with both Virtex II and Virtex 4. Though both versions could initially program any of these Virtex family FPGAs, the original Virtex requires a Selectmap abort sequence to be issued before new configuration data can be loaded into the FPGA without power cycling the FPGA. The newer Virtex

generations contain a command in the configuration data stream that stops synchronization with the data stream after programming has been completed. This removes the need for an abort sequence in the newer FPGAs.

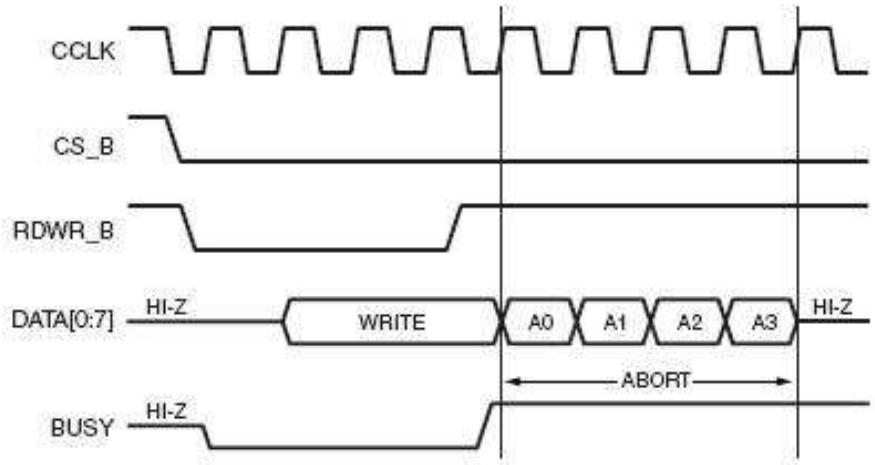

Figure 5·6: SelectMap Abort Sequence Waveform [8]

The Selectmap interfaces can program using data from flash or data sent from the host PC, and continuously reprogram from the flash memory at a hardcoded time interval. Below is a list of the available commands sent from the host pc through an ISA Bus interface and their functions:

- **Load Start Address:** The host PC sends 0x0A followed by three additional bytes containing the starting flash address where a given configuration data stream is stored. This address is stored inside the Selectmap interface.
- **Load Stop Address:** The host PC sends 0x01 followed by three additional bytes containing the last flash address where a given configuration data stream is stored. This address is stored inside the Selectmap interface.
- **Program using Flash:** The host PC sends 0x02 to initiate the Selectmap interface to send a data stream from the Flash memory to the FPGA configuration controller. The Selectmap interface returns a 0xBF (Begin Flash) to the ISA interface and a

0xEF (End Flash) when the data transfer has been completed.

- **Load Bus Word Number:** The host PC sends 0x03 followed by 4 bytes that represent a 32-bit number representing the total number of 8-bit words to be sent from the host PC to the FPGA configuration controller across the Selectmap interface. This value is stored in the Selectmap interface once transferred.

- **Program using Bus:** The host PC sends 0x04 to start a FPGA configuration programming. A 0xBB (Begin Bus) is sent to the host PC as a command acknowledge. When the final word is transferred to the Selectmap interface, it returns a 0xEB (End Bus) to acknowledge the completion of the configuration data transfer.

- **Scrub using Flash:** When a 0x05 is sent the Selectmap interface will begin continuously sending a configuration data stream located between the start and stop flash addresses previously loaded. A pause interval between transfer is coded into the VHDL. Interval times are also affected by the availability of the Flash interface, which is controlled by the Flash arbitrator. In the first generation Virtex Selectmap interface version an abort sequence is issued before each scrub iteration. An acknowledge is sent to the host PC when the command begins execution. The interface monitors for a stop command (0x0E) from the host PC to stop the execution. When received the last cycle is finished before sending a "scrubbing stopped" acknowledge back to the host PC. This command and function is necessary for fault mitigation strategies discussed in FPGA Fault Mitigation Techniques Overview section and in depth in later sections

- **Abort:** This command, 0x08, is only available in the first generation Virtex Selectmap interface. It is used to cause a selectmap abort before any new configuration data can be sent to the FPGA configuration controller.

5.1.6 Inter-FPGA Communication Interface

This interface is a modified ISA Bus interface for use in giving any co-processing designs instantiated in a second FPGA address space on the Bus for direct access to and from the host PC. The modifications were necessary to allow reliable transfers where delays across pins were now an issue. The modifications were made based on the logic analysis observations made possible by Chipscope Pro. There is no consideration for crossing clock domains as the NRL board uses the same physical clock connection for both FPGAs. This interface was designed specifically to use the resources of the support FPGA and use minimal resources on the co-processing FPGA.

5.2 Host PC Support System Software

This software is compiled for use on the Linux Operating System installed on an x86 based PC. In particular, they have been tested with the Slackware Linux distribution running on Tri-M Engineering's TMZ104 embedded x86 processor. The programs perform the functions needed to send data and commands from the Host PC and other outside sources to the FPGA Support System.

5.2.1 Flash Program

As the title suggest this program sends a bitstream file containing a configuration data stream to the support FPGA system for programming to the Flash memory. It is executed from a shell with the command "fprog bitsreamfile". The program loads the file into RAM, counts the number of 16-bit words in the file, and sends the program flash command and number of words to the Flash control interface. As the data is streamed across the Bus to the Flash control the FPGA input FIFO status is monitored so that no data overflow occurs.

5.2.2 Flash Verify

The program reads back the data contained in Flash memory and compares it to the version of the file located on host PC hard drive. It is executed using "frdback bitstreamfile". If an error results the program displays the address in flash where the error occurred and what the expected value was.

5.2.3 Selectmap Configuration

This program either transfers a configuration data stream across the bus or instructs the support system to program the coprocessing FPGA. The program monitors the FPGA input FIFO flags in the ISA Bus interface to prevent data overflow. It is executed using "smprog bitstreamfile". The bit stream file is only necessary if performing a configuration using data coming from the bus. Acknowledges sent back from the support system to the program are used to notify user of successful execution.

5.2.4 Scrub On and Scrub off

Executing these commands instructs the Selectmap interface in the support system to start or stop scrubbing the coprocessing FPGA. Each waits for an acknowledge from the support system and displays the information to the user. With an appropriate configuration data stream loaded into Flash memory these programs and the support system can repair SEUs in the configuration memory.

5.2.5 Co-processing Proof of Concept Program

A co-processing FPGA design was created to increment any value received and send back the incremented value. This program verified correct operation of the inter-FPGA communication module by sending burst of values to the coprocessing FPGA and checking the received values returned.

Chapter 6

Support System Fault Mitigation

Fault tolerance for commercial off the shelf FPGAs has previously been described as having two fronts. Correcting SEUs in an instantiated design's architecture while simultaneously using fault mitigation techniques inside the design to allow continuous error free service. For the developed support system used in the NRL board to be a viable technology for SEU inducing radiation environments both the configuration memory in the support FPGA and the configuration memory in the co-processing FPGA must be scrubbed to correct upsets. In short, the support system must be able to correct its own architecture as well as any other modules' architecture instantiated on other FPGAs. The support modules must also be translated into fault tolerant designs using various mitigation techniques.

The immediately preceding subsection will first describe the details of initial configuration, active reconfiguration, and scrubbing. This includes how one FPGA can scrub itself and any other FPGAs configuration memory and the configuration data stream specifics to accomplish these tasks. The remaining subsections will detail how the HDL modules were made fault tolerant. These sections will first cover techniques common to all modules and then techniques specific to certain areas.

6.1 FPGA Configuration, Scrubbing and BitStream manipulation

There are three FPGA configuration modes needed to achieve the support system functionality: initial configuration, reconfiguration, and active partial reconfiguration. From the perspective of the Selectmap interfaces that send the configuration data streams to the FPGA configuration controller there is no difference between any of these three modes. However, the commands and data contained with in the configuration data streams is

where the differences occur and have drastically different effects on chip operation. The original files that contain the configuration data streams are referred to as bit files and contain bit streams. The bit streams are broken up into 32-bit words where each command is 32-bit longs. An initial configuration BitStream contains a series of setup commands, the configuration data to program the desired functionality into the FPGA, and lastly a series of startup commands to bring the FPGA into service.

A reconfiguration BitStream is the same as an initial BitStream except it contains additional commands to first shutdown the FPGA and bring it out of service. This allows new a design to be instantiated into a FPGA without having to power cycle the FPGA or assert the program pin that clears the entire configuration memory. The series of commands shown below is issued over the BUS by the SelectMap interface before programming a new design to the co-processing FPGA.

Configuration Data	Explanation
30008001	Type 1 write 1 words to CMD
0000000B	SHUTDOWN command

Figure 6·1: SelectMap Shutdown Command Sequence [8]

An active partial reconfiguration BitStream is used for Fault Mitigation of the FPGA configuration memory. It is a bitstream that contains only certain portions of the configuration data and no start up or shutdown commands. This allows for the safe repairing of configuration memory faults while the FPGA is still active and providing service. The portions of the BitStream responsible for BRAM initial values are removed while leaving the portions responsible for the BRAM interconnects and all other portions behind. This allows modules with dynamic data in the BRAMs to continue to be accessed without interruption and the remaining architecture to be cleared of any faults from SEUs.

All the BitStreams are created by the Xilinx Bitgen tool. Executing the BitGen tool from a command line one can create active partial reconfiguration BitStreams by masking

out unwanted the portions of the BitStream. The commands used to create active partial reconfigurations with BRAM inital data removed are:

```
Virtex II 6000 Bitgen Command Example

bitgen -g activereconfig:yes -g binary:yes -g persist:yes
-gpartialmask0:1FFFFFFFFFFFFFFFFFFFFFFF -g partialmask1:00
-g partialmask2:3F <design.ncd>
```

For the support system used in the NRL Board, programming the configuration memory of both FPGAs is accomplished by using two copies of the Selectmap interface module. One copy of the newer Virtex generation module is attached to the configuration pins of the co-processing Virtex II FPGA. The second copy of the module is connected to the Selectmap configuration pins of the support Virtex FPGA by routing through the co-processing FPGA. The extra routing step was the only way to physically connect to the Selectmap configuration pins on the support FPGA. If a direct connection existed there would be no need for the extra routing step.

The initial configuration for the support FPGA is stored on an EEPROM for powerup. The initial co-processing configuration, active partial scrubbing co-processing configuration, and the scrubbing support configuration BitStreams are all stored at pre-designated locations in the flash memory. The Selectmap interfaces when given scrubbing commands use the data stored in the Flash memory to periodically write the scrubbing BitStreams to both FPGAs configuration controllers and repair any configuration memory upsets that may have occurred to either FPGA.

6.2 Mitigation Techniques Common to All Modules

6.2.1 Triple Module Redudancy

The primary technique used to provide continuous service from an FPGA is Triple Module Redunancy (TMR). Triplication of individual HDL modules allows for continued service when configuration faults cause one of the three copies to function incorrectly. A majority voter determines the output of the three identical modules. This technique is covered in

numerous academic papers, however a good summary and detailed explanation of how to implement the designs can be read in Xilinx Application Note 197, Triple Modular Redundancy Design Techniques by Carl Carmichael [9]. When the output of the three modules does not have to be sent to a single pin or HDL module port three copies of the voter circuit should also be instantiated. This would result in three majority voted output signals to be sent to the next TMR HDL module. This adds further area redundancy incase an upset occurred in the configuration memory along one of the triplicate signal paths.

One might raise the issue of an upset occurring within a majority voter itself. If a majority voter were made in the same manner as all the other logic within a normal design errors to a majority voter would result in incorrect or discontinued operation. However, using tri-state buffers that are located throughout Xilinx FPGAs one can create a majority voter that is not susceptible to upsets in the same manner. The tri-state buffers are not

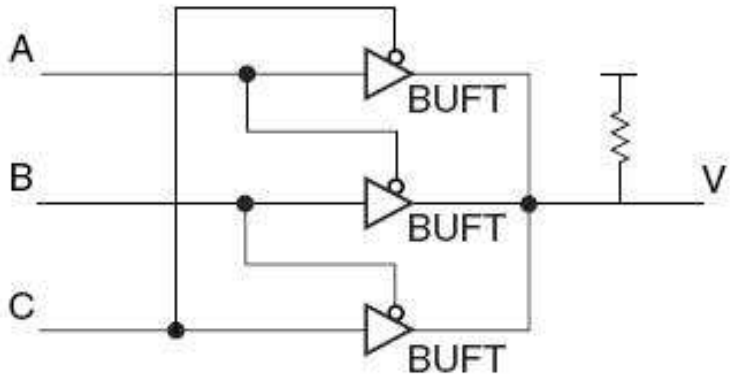

Figure 6·2: Tri-State Buffer Majority Voter [9]

made of the same SRAM material used throughout the rest of the FPGA. In fact, only the interconnect structure connecting the tri-state buffers is susceptible to upset. Even if a disconnect occurs the correct majority voting function would still result. A disconnected tri-state buffer will no longer drive its output, but the final voter output will not be affected. [13] It would take multiple upsets to the majority voter structure within a single scrub cycle for a functional fault to occur. [13][9] In simulation the tri-state voters will

function correctly if no simulation SEUs are injected. Incorrect simulated operation will result if attempts to test the voters by giving different inputs are attempted. [13][9]

A second issue when utilizing TMR is the small possibility of an SEU occurring in the routing structure and affecting the operation of two of the three TMR modules. An upset causing design routing errors in both modules by connecting or otherwise causing them to interfere with one another can result in a majority voter picking the incorrect value. Mitigation of this small possibility is obtained by module placement during the place and route portion of FPGA design creation. Xilinx Synthesis Tools allow for the ability to constrain modules to certain physical portions of the chip. Placing the triplicates of a module in different locations can remove this error. However, one must balance the distance between placement as this generates larger signal paths for each module that can be exposed to SEUs by increasing the routing needed to bring the individual outputs to the majority voters. Each module much contain the following two lines in its VHDL entity portion of the source code before synthesis occurs [14]:

```
attribute keep_hierarchy: string;
attribute keep_hierarchy of sm_control: entity is "yes";
```

This causes the synthesis tool to build the synthesized design with module hierarchy and allows the Xilinx Place and Route Constraints Editor (PACE) to recognize individual modules in the synthesis file. The PACE GUI can then be utilized to constrain the different modules to specific locations that are saved in the user constraints file. The user constraints file is then referenced during the place and route portion of a VHDL design implementation to meet the desired layout.

6.2.2 Finite State Machines and TMR

Finite State Machines (FSMs) are properly structured and used in almost every module in the support system. FSMs can not simply be triplicated like other modules. An upset in one of the FSMs would cause it to permanently loose synchronization with the other

copies until a reset to bring all three back to the same state was issued. FSMs contain a combinational logic network using the current state and inputs to determine the current outputs and next state. A visual picture of how to use TMR and retain synchronization is shown in FIGURE 6.3. The next state value to be stored in each state register copy

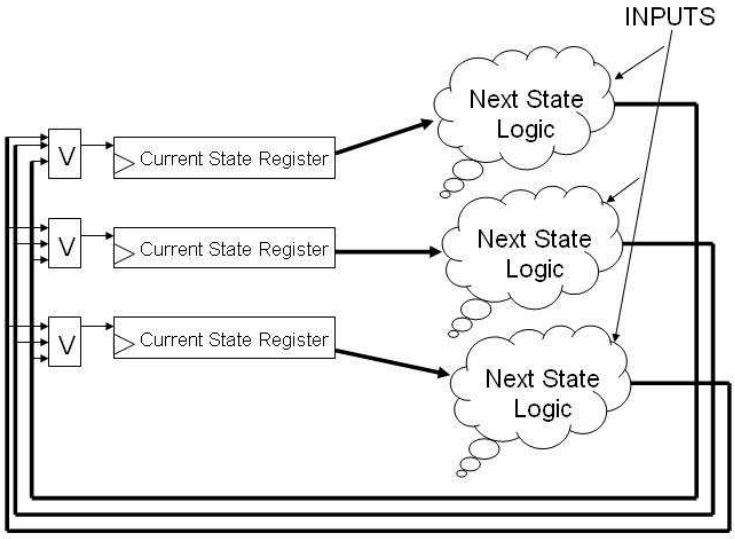

Figure 6·3: Finite State Machine TMR Implementation

is majority voted before being latched resulting in synchronization. This prevents Single Event Transients in the combination logic from becoming SEUs that would cause one of the FSMs to lose synchronization with the others. If an upset occurred in the state register the incorrect next state and outputs would be voted out and the correct state would be latched into the register on the next clock cycle. The output from each FSM is also majority voted before leaving the module.

6.2.3 TMR for Xilinx Block RAMs

Xilinx Block RAMs are susceptible to upsets and need the addition of fault mitigation techniques. Scrubbing protects the routing structure connecting the BRAMs to the rest of a design, but triplication of the BRAMs and the routing structure is necessary to allow continuous service until the SEU in the routing structure is repaired. This necessary repli-

cation also serves as error correction to mitigate upsets to the BRAMs stored data values by majority voting the correct result from the three BRAMs. Simple redundant TMR designs were originally created that simply allowed upsets to accumulate in individual BRAMs and the outputs were voted out. This was considered acceptable for most applications, as the data in most cases would not be stored for extended periods of time. Data storage periods were in most cases much less than the expected SEU rates.

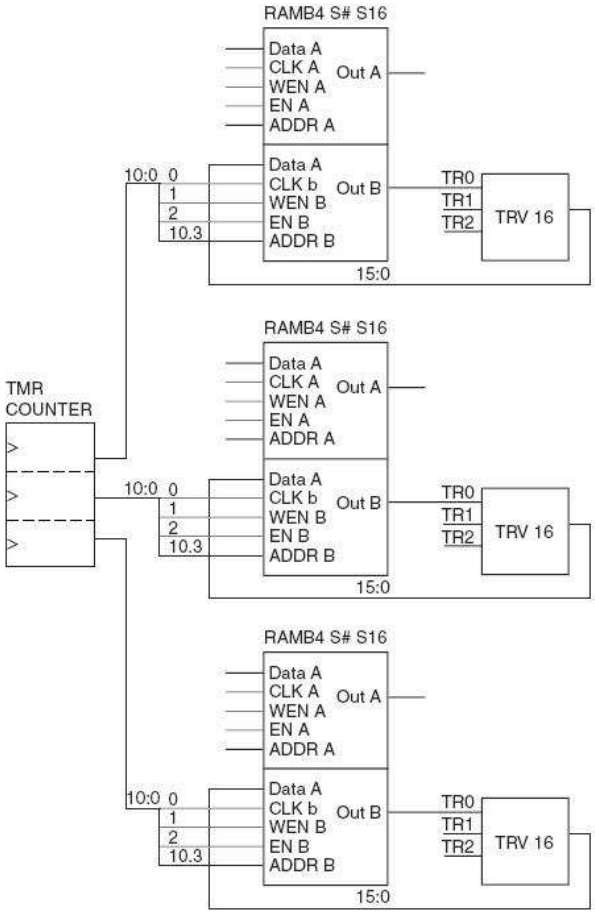

Figure 6·4: TMR BRAM with Refresh [9]

The Xilinx Application Note 197 [9] included a link to designs examples of TMR BRAMs on the Xilinx support FTP server. These TMR BRAM designs included simple redundancy combined with refresh. Since, all BRAMs are dual ported the second port is utilized

to continually refresh stored data values removing upsets. Refreshing works by cycling through all BRAM addresses, majority voting the results, and then storing the voted result back to all three BRAMs. Three copies of the majority voter are used.

These TMR BRAM designs also include data collision detection. Collision detection prevents the refreshing of a stored data value at a given address if that address is being written to during the same clock cycle. This prevents the possibility of new data values being replaced by old ones by the refresh circuitry.

6.3 Specialized Fault Mitigation Techniques

These techniques primarily deal with taking data on and off the FPGA chip and were issues that needed to be specifically dealt with to use the support system on the NRL dual FPGA board. These issues arose from the physical restriction of available pins for TMR across physical devices or because of only one physical device being available to interface with. All of the interfaces described below require off chip communication.

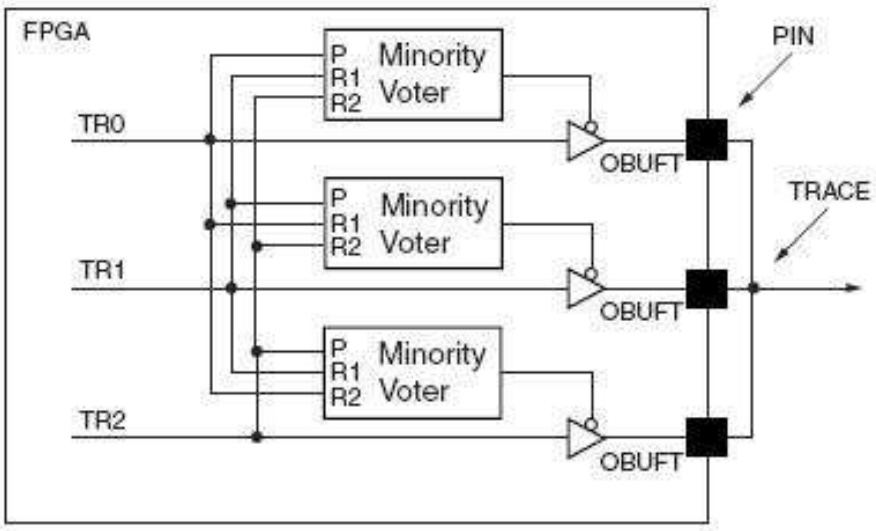

Figure 6·5: TMR Package Pin Minority Voter [9]

The NRL board did not have three physical pin connections to each off chip signal. If the physical connection existed the techniques described below would not be necessary.

Inputs would simply be read on the three pins and then majority voted to correct upsets to a pin caused by Single Event Transients or configuration memory errors. Outputs would use a design referenced in source [9]. A pin with a minority value would go into high impedance mode and not affect the correct signal leaving the chip to the Bus.

6.3.1 Flash Interface Mitigation Techniques

Flash chips are impervious to SEUs in LEO orbit. However, the Finite State Machine inside these chips and the FPGA pins connecting to the Flash pins are not. Upsets to the configuration memory of the FPGA can cause pins to function improperly yielding incorrect data reads and writes. Upsets to the FSM inside the Flash memory in most cases will result in incorrect operation that will eventually return to a normal state where it can be accessed again. However, in some cases these faults can only be corrected by a power cycle or can cause permanent device failure.

Errors during writing to the Flash memory are corrected by simple verification through read back and comparison and then rewriting the stored data when an error occurs. This is handle by the host PC and software programs previously described with these functions. This is the only strategy necessary as currently only the host PC will change the data in the Flash memory.

Faults that occur during reads from the Flash memory are corrected in a more complex manner. Since data stored in the flash is known to be free of errors, the only source of errors is the communication pathway between the Flash and FPGA. A Cyclic Redundancy Check is used to verify the correct transfer of data. During Flash programming the host program embeds a CRC16 value calculated for every 512 16-bit flash data words into the stored data. 512 words are read off the Flash memory into an internal FPGA BRAM buffer. A CRC16 module calculates a result and compares it to the stored value in the flash memory. If the value matches the buffer is used for flash data requests in that address range from modules. If the CRC value does not match the buffer is cleared and the data is read again after a time period greater than the scrubbing period for the given FPGA. The

Flash data buffer can be thought of as an on chip Flash memory cache for the other HDL modules when verified. The buffer is of course also protected using TMR. This strategy could also be used for ROMs or other devices where the stored data is known to be fault free, but the communication pathway to the FPGA modules is not.

6.3.2 Inter-FPGA Communication and Bus Interface Mitigation Techniques

The number of pins available for communication between the two FPGAs is too small in number to allow triplication and majority voting of all signals. Otherwise, all signals would be triplicated across three pins and then voted on the receiving end. The Bus does not have triplicate physical pins connections to allow TMR for bus data transfers. An 8 data bit and 4 redundant bit Hamming code can solve both of these issues. This Error Correction Code allows for single error correction and double error detecting.

The Bus interface was updated to use the extended ISA data transfer mode that allows the transfer of 16 bits during a single Bus transaction. The Host PC encodes every byte sent across the Bus, it is then decoded and if necessary corrected by the Bus interface and stores the byte in the input FIFO. The inverse occurs for data leaving the Bus interface FIFO. A value is taken from the FPGA output FIFO, encoded, and sent across the Bus where the host PC decodes and corrects the returned byte. However, this does not protect individual single bit signals from upsets. Only allocating three package pins to each of these Bus signals could remove the cross section pertaining to these signals.

For byte transactions occurring during inter-FPGA communication the (8,4) Hamming Code is used on the data bytes. This took the number of pins needed just for the data from 48 using TMR to 24 using the Hamming Code. All single bit signals, such as read and write signals, are still used in triplicate.

6.3.3 SelectMap Interface Fault Mitigation

As described in the beginning of specialized mitigation techniques section, off chip communication can use TMR by having three pins sample the same original signal and outputs

using the minority voter pin removal design shown in figure 18. Though the physical ability to utilize these techniques is not available on the current version of the NRL board, it is the only way possible to offer fault tolerance for this application. It is worth noting that future boards should use this mitigation technique to further lower the support system design cross section. Errors in communication can cause SEFIs as a result of incorrect faults occurring to the FPGA configuration controller and memory.

6.4 Host PC Fault Tolerant Support System Software

The changes to the programs mentioned in the previous software section were touched upon in the descriptions of the new fault tolerant modules. Five final programs are now necessary to work with the fault tolerant support system.

New flash program and verify programs now program and verify all the needed BitStreams at once. These BitStreams include the initial coprocessing, the scrubbing coprocessing, and the scrubbing support BitStreams. Both programs also create, embed, and verify the CRC words for every 512 16-bit values. All bus transactions sent from and received by the flash program are also now encoded using the (8,6) Hamming Code.

The program responsible for programming the co-processing FPGA has also been updated beyond using the Hamming Code for all Bus transactions. Pre-defined Flash address values are sent to program the target FPGA defining the location of the initial configuration BitStream. The ability to program the co-processing address using a configuration BitStream transferred over the Bus is still available.

The new scrubbing on and off programs have also been updated to use the (8,6) Hamming Code to encode and decode all Bus transactions. The scrubbing programs now interact with both the support and co-processing SelectMap interface modules. They have been changed to send pre-defined Flash addresses marking the beginning and ending locations for both the support and co-processing FPGA scrubbing BitStreams.

Chapter 7

The End User Environment view of the ToolKit

The End User of the tool kit is expected to have a working knowledge of both VHDL and C++. This user will use the tool kit to develop co-processing algorithms with fault mitigation being a secondary and not primary design consideration. The final result of work using the toolkit will be a Fault Tolerant FPGA co-processing system where users simply send data and request results from C++ functions or already compiled programs. To help facilitate end users in reaching this final result the toolkit offers several options.

The Fault Tolerant Modular Support System offers end users the ability to use the system on boards similar or identical to the NRL board with little to no changes necessary. The modular design allows for new components to be quickly integrated and others removed if required by the end user while leaving the majority of the VHDL code in the same form.

To help in the creation or translation of existing co-processor designs VHDL examples, templates, and generic components for fault mitigation techniques are included. A FSM machine template is included allowing quick creation of TMR versions. The user needs to only "look" at the FSM machine as a single entity while the template is used to perform synchronization and TMR. A VHDL majority tri-state voter was created with a feature that allows changing a value in the VHDL source code that will result in synthesis of a voter for whatever port bit sizes are needed for the module being triplicated.

C++ header and source code files are also included along with the previously described programs to be used in larger end user programs. The header files and source code include functions to program the flash, start and stop scrubbing, and transfer data on and off the coprocessor FPGA.

Chapter 8

Conclusion

The Fault Tolerant FPGA Co-Processing Toolkit provides a support system to take care of all underlying fault mitigation and data routing tasks. Simply reusing support modules a support system can be constructed for a single to multiple FPGA co-processing system. The toolkit can currently be used for Xilinx Virtex FPGAs including the new Virtex 4.

End users will exploit the toolkit for use in creating new co-processing algorithms, support HDL modules, and embedded host PC software for new FPGA based PCBs. Future support and co-processing modules currently underdevelopment include a DRAM interface and a Fast Fourier Transform algorithm implementation. As the toolkit grows, its overall strength as a useful tool for more applications and new generations of physical boards will grow. The overall goal is for the toolkit to help facilitate further acceptance of FPGAs for use in processing needs and its eventual acceptance as a required tool in SEU inducing radiation space environments for performance co-processing applications.

References

[1] Tom Van Court, Martin C. Herbordt, and Richard J. Barton, "Microarray data analysis using an FPGA-based coprocessor," Microprocessors and Microsystems 28, 4 2004, 213 - 222

[2] Yongfeng Gu, Tom Van Court, Douglas DiSabello, and Martin C. Herbordt, "Preliminary Report: FPGA Acceleration of Molecular Dynamics Computations," Field-Programmable Custom Computing Machines, 4 2005, 269 - 270

[3] Xilinx Inc., "Virtex Series Data Sheet," 2001

[4] Tanay Karnik Peter Hazucha, Jagdish Patel, "Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes," IEEE Transactions on Dependable and Secure Computing, 1, 2, 2004

[5] K.L. Bedingfield, and M.B. Alexander, "Spacecraft System Failures and Anomalies Attributed to the Natural Space Environment," National Aeronautics and Space Administration Reference Publication 1390, 1996

[6] Digital Engineering Institute, "Space Radiation Definitions,"
http://klabs.org/richcontent/Tutorial/Radiation Definitions.htm
accessed: March 2006

[7] Michael Affrey, Paul Graham, Eric Johnson, Michael Wirthlin, Nathan Rollins, and Carl Carmichael "Single-Event Upsets in SRAM FPGAs," Military and Aeropspace of Programmable Logic Devices, September 2002

[8] Xilinx Inc., "Xilinx Inc. Virtex II Series Configuration Architecture User Guide," 2005

[9] Carl Carmichael, "Triple Modular Redundancy Design Techniques for Virtex FPGAs," Xilinx Application Note 197, 2001

[10] D.N. Nguyen, S.M. Guertin, G.M. Swift, and A. H. Johnston, "Radiation Effects on Advanced Flash Memories," IEEE Transactions on Nuclear Science, 46 (6), 1744 (1999)

[11] James C. Coudeyras, "Radiation Testing of The Configurable Fault Tolerant Processor (CFTP) for Space-Based Applications," Thesis, United States Naval Post Graduate School, 2005

[12] Intel Corp., "Intel Advanced+ Boot BLock Flash Memory (C3) Datasheet," 2005

[13] Carl Carmichael, Earl Fuller, Phil Blain, Michael Caffrey, "SEU Mitigation Techniques for Virtex FPGAs in Sace Applications," 2003

[14] Xilinx Inc. "Xilinx Synthesis Tools User Guide," 2005

[15] Frenanda L. Kastensmidt, Gustavo Neuberger, Luigi Carro, Ricardo Reis, "Designing and Testing Fault-Tolerant Techniques for SRAM-based FPGAs," Computing Frontiers 2004, April 14-16, 2004

[16] Dean A. Ebert, Charles Hulme, Hershel Loomis, and Alan A. Ross, "Configurable Fault-Tolerant Processor (CFTP) for Space Based Applications," 17th Annual AAIA/USU Confence on Small Satellites 2003

CURRICULUM VITAE

DOUGLAS MICHAEL DISABELLO

Candidate for MS Computer Systems Engineering May 2006

BS Electrical Engineering, Boston University 2004

Graduate Research

Awarded grant from United States Naval Research Lab
Design and implementation of configurable fault tolerant computing systems for high radiation space environments utilizing off the shelf FPGAs and custom embedded systems. These will be exploited to bring high performance computation normally only available from terrestrial systems to orbiting space systems.

Work Experience
Summer Intern Electrical Engineer, Naval Space Sciences Division
United States Naval Research Lab, Washington, DC (May 2005 August 2005)

Research Assistant, Computer Architecture and Automated Design Lab
Boston University, Boston, MA (September 2004 Present)

Research Assistant, Advanced Materials Process Control Lab
Boston University, Boston, MA (June 2000 August 2002)

Teaching Assistant, Computer Organization (ENG SC312)
Boston University, Boston MA (September 2003 May 2004)

Publications
Preliminary Report: FPGA Acceleration of Molecular Dynamics Computations
Yongfeng Gu, Tom Van Court, Douglas DiSabello, Martin Herbordt

Awards
Boston University Scarlet Key Award and Honor Society Induction
Undergraduate Research Opportunity Program Faculty Matching Grant