Spatial Avoidance of Hardware Faults using FPGA Partial Reconfiguration of Tile-Based Soft Processors

Clint Gauer
406-994-2505
cgauer@montana.edu

Brock J. LaMeres
406-994-5987
lameres@ece.montana.edu

David Racek
406-994-2505
david.racek@montana.edu

Electrical & Computer Engineering Department, Montana State University, Bozeman, MT 59717

Abstract—This paper presents the design of a many-core computer architecture with fault detection and recovery using partial reconfiguration of an FPGA. The FPGA fabric is partitioned into tiles which contain homogenous soft processors. At any given time, three processors are configured in triple modulo redundancy to detect faults. Spare processors are brought online to replace faulted tiles in real time. A recovery procedure involving partial reconfiguration is used to repair faulted tiles. This type of approach has the advantage of recovering from faults in both the circuit fabric and the configuration RAM of an FPGA in addition to spatially avoiding permanently damaged regions of the chip. [1] [2]

Table of Contents

1. Introduction

When cosmic particles (typically heavy ions and protons) strike integrated circuits (IC), fault conditions called Single Event Effects (SEE) can occur [1]. The particles ionize the semiconductor material used in the circuit causing a variety of fault conditions. Single Event Transients (SETs) occur when the electron/hole recombination in the ionized material causes a voltage spike on the output of the device. When the magnitude of the SET is large enough to cause a logic transition on a receiving gate, logic failures in the circuit can exist [2-3]. A Single Event Upset (SEU) refers to when an inadvertent logic transition is captured in a digital storage device such as a flip-flop or SRAM cell [4]. When an SEU occurs in the logic fabric of an FPGA, it is referred to as a *soft fault* because no permanent damage is caused in the circuit and the fault can typically be recovered using a reset [5]. When an SEU occurs in the configuration RAM of an active region of an FPGA, it is referred to as a Single Event Functional Interrupt (SEFI) because a simple reset will not restore the initial state of the circuit [6]. This type of failure does not cause permanent damage to the FPGA; however, traditional reset and recovery sequences cannot be used since the physical circuitry in the FPGA fabric has been altered.

There has been a significant amount of work on mitigating SEUs in digital circuits. Triple modulo redundancy (TMR) has been widely adopted as a way to detect and correct logical errors by using three redundant circuits and a voter [7]. The voter circuit produces an output dependant on the majority of outputs from the three circuits. For more complex systems, TMR can be used in conjunction with a recovery sequence which can reset and reinitialize the system when a fault is detected [8]. Similar techniques of using redundancy have been applied at the application layer. These techniques (known as *radiation hardened by software)* are based on implementing TMR by running redundant processes and performing the voting in the software [9]. Watchdog timers have also been deployed broadly as a fault mitigation technique. Watchdog timers independently observe the operation of a system and initiate a reset when the system becomes idle for too long [8]. These types of logical solutions are easily adopted in reprogrammable fabrics. The downside of these approaches comes in the form of increased area, reduced performance, and additional power consumption. Furthermore, finding the optimal fault observation nodes is a challenge due to the impact the observation circuitry has on the operation of the circuit being monitored [5-6]. Another challenge is detecting faults that occur in the checking circuitry. Despite these challenges, logical fault detection and recovery techniques have been deployed broadly in military and aerospace systems, particularly when the systems contain FPGAs.

One of the biggest challenges in using FPGAs in aerospace applications is the susceptibility of the configuration RAM to SEUs [10-11]. Typically, FPGAs that are used in high radiation environments have a fuse-based configuration RAM. This avoids SEFIs, but limits the flexibility of the design. Fuse-based FPGAs do not have the ability to reconfigure in the field, which precludes some

1———————

[1]978-1-4244-3888-4/10/$25.00 ©2010 IEEE
[2] *IEEEAC paper#1079, Version 3, Updated 2009:10:23*
This research was carried out at Montana State University, Bozeman with funding from NASA EPSCoR and the Montana Space Grant Consortium (MSFC1-08-SD)

of the attractive options that non fuse-based FPGAs have such as reconfigurable computing. In FPGAs that have SRAM-based reconfiguration memory, a *scrubber* circuit is typically used to detect and correct SEUs in the configuration RAM (i.e., SEFIs). A scrubber is a circuit which continually compares the data in the configuration SRAM to the original configuration data that resides in an off-chip non-volatile device [12]. When a scrubber detects an error, it overwrites the reconfiguration SRAM with the original values. The advantage of adopting a scrubbing technique is that it runs independently of the main system hardware so it does not require integration into the main circuit. The drawback of traditional scrubbers is that they don't have insight into where in the configuration SRAM an SEU might have occurred. They simply traverse through the memory addresses checking the contents. This can lead to a significant latency between the detection and repair of a configuration SRAM SEU.

There are also a number of radiation effects that can cause permanent damage to FPGAs. Total Ionizing Dose (TID) refers to the long term damage to a device mainly due to low energy electrons and protons [13]. TID effects result from charge carriers getting trapped in the insulating or more lightly doped regions of the device. When an electron/hole pair is created by the radiation strike, the carriers attempt to move back together to find an electrostatic equilibrium. The electron and hole charge carriers experience different mobility rates due to the properties of the materials they pass through. Electron mobility (μ_n) of semiconductor materials tends to be higher than hole mobility ($\mu_n > \mu_p$). As a result, the hole charge carriers have a higher likelihood of getting trapped within the insulating or more lightly doped regions of the device due to the increased time it takes for them to recombine. This phenomenon permanently degrades the transistor and can result in threshold shifts, increased device leakage, timing changes, and ultimately functional failure of the device [1].

A number of solutions have been developed to increase a part's resilience to long term TID exposure. Techniques such as isolation trenches, substrate doping, and using non-standard layout techniques are just a few examples of approaches that have been used to make integrated circuits radiation hardened [14]. Parts that have been radiation hardened are specified to withstand a particular dosage (typically >300krad). The primary drawback of these techniques is that they require a dedicated radiation hardened process to perform the fabrication. This leads to increased cost of the devices. Furthermore, the fabrication techniques used decreases the performance of the devices compared to commercially fabricated parts. This gap in performance leads to a number of issues including hardware and software compatibility with emerging technology [15]. TID hardened parts do not prevent SEUs caused by high energy particles so logical mitigation techniques are still required.

The inherent flexibility and increased performance of SRAM-based FPGAs has spurred great interest from the aerospace community to evaluate their usage in flight systems [16-17]. As these parts are considered, novel fault detection and recovery techniques must be developed that can mitigate radiation induced errors. Recent advances in the development tools for FPGAs have enabled direct access to the configuration SRAM. This has allowed techniques such as partial reconfiguration to be used as a fault mitigation approach [18].

In this paper, we present a tile-based, soft processor computing system. In this approach, an FPGA is divided into equally sized *tiles* which represent a quantum of resources that can implement a *Xilinx picoBlaze* [19] soft processor ***and*** can also be individually reprogrammed using partial reconfiguration (PR). At any given time, three of the processors are configured in TMR with the rest reserved as spare processor tiles. In the event that the TMR voter detects a fault, a recovery process is initiated that will attempt to reset, reinitialize, and resynchronize the faulted tile. This recovery process mitigates SEUs that may have occurred in the FPGA circuit fabric. If the tile reset is not successful, a spare processor is brought online from one of the unused tiles to replace the faulted circuit. Once the new TMR triplet is operational, an attempt is made to recover the previously faulted tile using partial reconfiguration. After PR, the recovered tile is reintroduced into the system as an available spare. This recovery process mitigates SEUs that may have occurred in the configuration SRAM of the FPGA (i.e., SEFIs). If the system tries to use the recovered tile for a second time and immediately experiences a fault, the tile is marked as *permanently TID damaged* and is no longer available for use. This allows the system to continue operation in the presence of TID failures in localized regions of the FPGA.

The mitigation strategy we present in this paper has the advantage of addressing the two main logical fault types experienced in SRAM-based FPGAs (fabric SEUs and SEFIs). Furthermore, the ability to continue operation despite TID damage can extend the useful life of flight hardware. Our system was prototyped on a Xilinx Virtex-5 LX110 FPGA. This paper presents the design, implementation, and parametric results for our system. The paper begins with a description of the overall system architecture. Then the operation of the individual recovery modes are discussed (soft fault recovery, spatial avoidance of TID damage, and SEFI recovery using PR). Then the recovery time is discussed for each of the mitigation techniques to evaluate the overhead associated with this type of fault detection and recovery scheme.

2. System Design

Our many-core system contains 16 homogenous tiles, each containing a *Xilinx picoBlaze* soft processor. Each of the active soft processors run identical software to control a set of basic peripherals. For this prototype, the peripherals consist of a PS2 keyboard, PS2 mouse, and liquid crystal display (LCD). The computing system continually monitors the input keyboard and mouse and writes the input values to the LCD screen. The computer system was implemented on a Xilinx XUPV5 evaluation board with a Virtex-5 LX110 FPGA.

The system routes the TMR observation nodes (instruction address, instruction data, and I/O) for all 16 processors to a TMR *data switchboard* circuit. This circuit handles routing the TMR observation nodes for only the three active processors into the voter circuit. After the TMR voter determines the majority output value, it sends the correct information back to the three active processors through a *data signal router*. The *TMR voter and recovery circuit* contains the necessary logic to handle switching which three processors are active in addition to resetting and reinitializing any of the processors. Any processor that is not active is held in reset to eliminate power consumption.

A graphical user interface (GUI) was developed to monitor which of the 16 soft processors were active at any given time. Soft faults in the FPGA fabric were injected into the system using push buttons on the evaluation board. SEFI faults were induced in the configuration SRAM using the GUI. The GUI also allowed the user to switch between an automatic partial reconfiguration of a faulted tile or a manual procedure.

Partial reconfiguration was managed using a separate *microBlaze* soft processor. This processor handled retrieving the reconfiguration data for each of the 16 tiles from an off-chip storage device through a *Xilinx System ACE* component and programming the tile using the ICAP port on the Virtex-5. A core was created from the *Xilinx Core Generator* called *HWICAP* which handles the timing interface between the *microBlaze* soft processor and the physical ICAP port.

Figure 1 shows a photo of the laboratory setup. Figure 2 shows the floor plan of the many-core system. The highlighted blocks represent a tile, which is the smallest amount of logic that can contain a *picoBlaze* processor and also be partially reconfigured (see section 6). Figure 3 shows the block diagram for our many-core system.

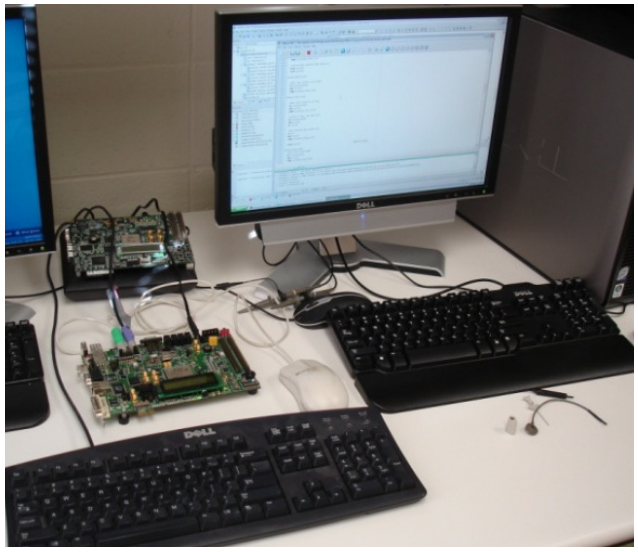


Fig. 1. Prototype system implemented on a Xilinx XUPV5 evaluation board.

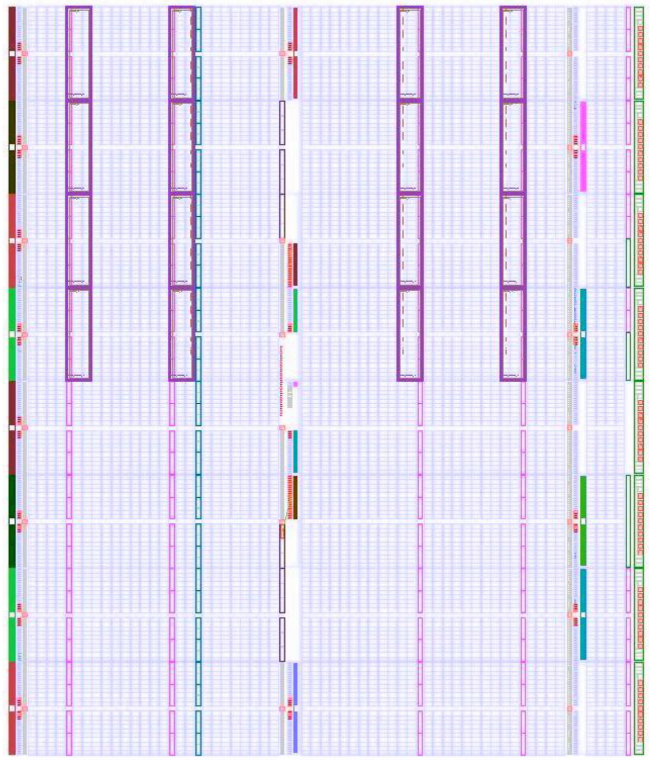


Fig. 2. Floor plan for the V5-LX110 FPGA highlighting the 16 reconfigurable tiles each containing a picoBlaze processor.

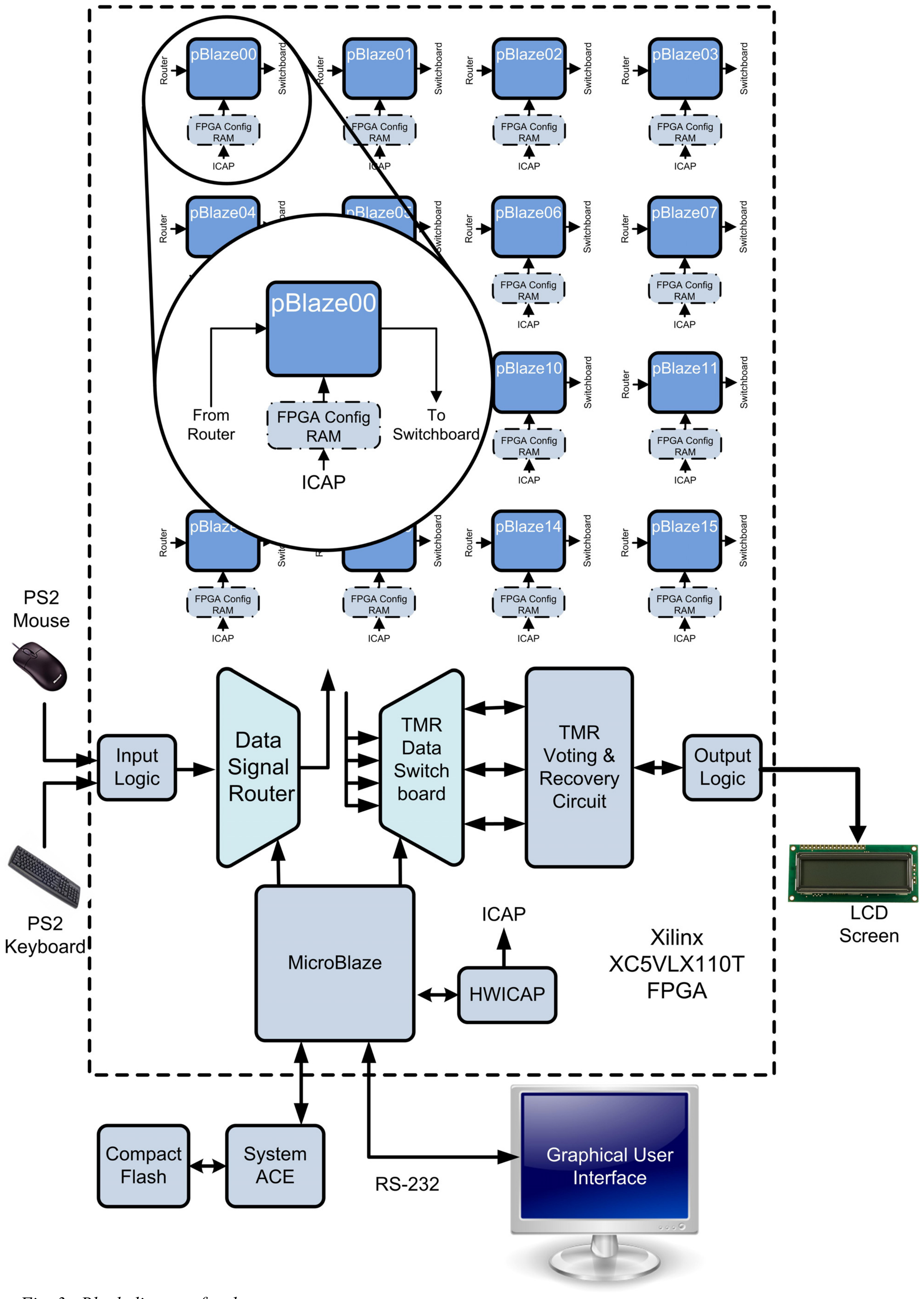


Fig. 3. Block diagram for the many-core system.

3. Triple Modulo Redundancy

A TMR voter and recovery circuit was implemented in VHDL to detect and correct faults in the soft processors. The TMR voter monitors the address and data lines between each microprocessor and its instruction memory. All of the observation nodes from the 16 soft processors are routed into the TMR voter. At any given time, three of the soft processors are active and are being voted on. The following figure shows the TMR block diagram for three active processors.

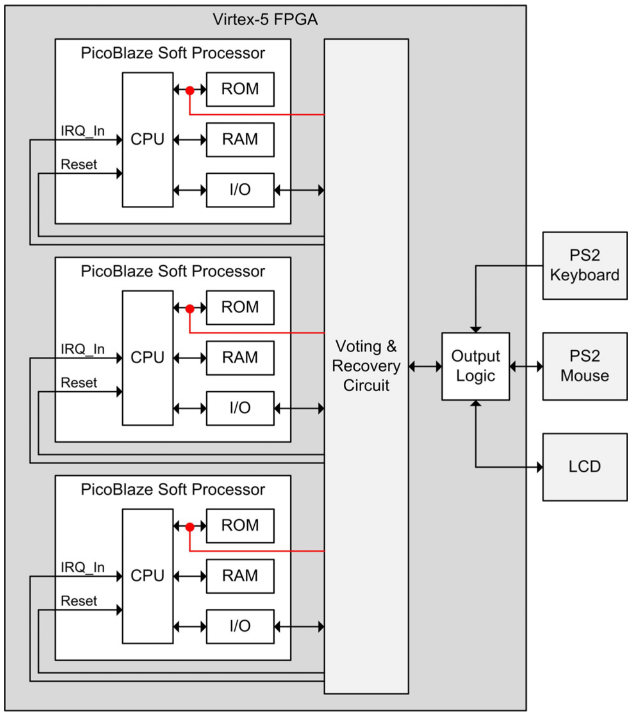


Fig. 4. TMR block diagram of three active processors.

4. Soft Fault Recovery

The TMR circuit monitors the instruction memory interface between the processor and memory. The TMR circuit also contains a state machine for the recovery of a faulted processor. When a fault is detected on one of the processors, the system attempts to recover the bad processor using a reset sequence.

Upon reset, each processor will read in its initial variable values from the TMR/recovery system. All processors are reset at the same time in order to ensure they are synchronized and initialized to the beginning of the main program loop.

The processors then enter their main program loop which services the peripherals. At the end of the main program loop, an *Error_Flag* is checked to see if a TMR failure has been detected by the recovery circuit. If no failure has been indicated, the computer continues to execute the main program loop.

In the event that a failure is detected by the TMR voter, the recovery state machine sends an interrupt to all processors. An interrupt service routine sets the *Error_Flag* indicating that a fault has been detected and processor recovery is necessary. When the main program loop checks the *Error_Flag* and sees it is asserted, it will then proceed to write all of its register and variable information to the TMR recovery system and then wait for a reset. In this manner, the processors will complete their current peripheral tasks prior to beginning the recovery sequence. The TMR voter ignores the variable data that is read from the faulted processor and stores the data from a good processor. The recovery state machine then resets all processors and reinitializes their variable data. The system operation and recovery sequence flow charts are shown in the following figure.

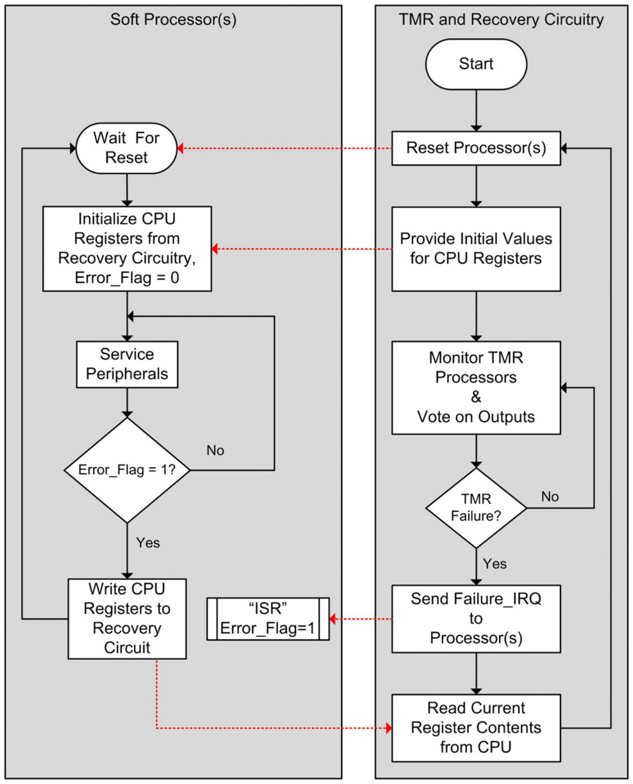
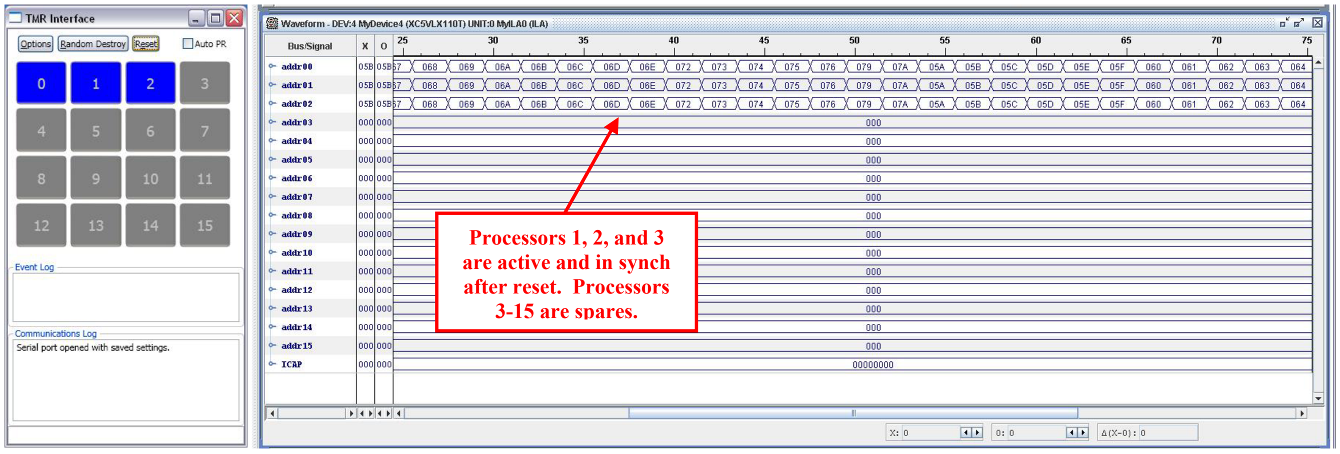

Soft Processor(s)
TMR and Recovery Circuitry
Start
Wait For Reset
Reset Processor(s)
Initialize CPU Registers from Recovery Circuitry, Error_Flag = 0
Provide Initial Values for CPU Registers
Service Peripherals
Monitor TMR Processors & Vote on Outputs
Error_Flag = 1?
No
Yes
TMR Failure?
No
Yes
Write CPU Registers to Recovery Circuit
"ISR" Error_Flag=1
Send Failure_IRQ to Processor(s)
Read Current Register Contents from CPU

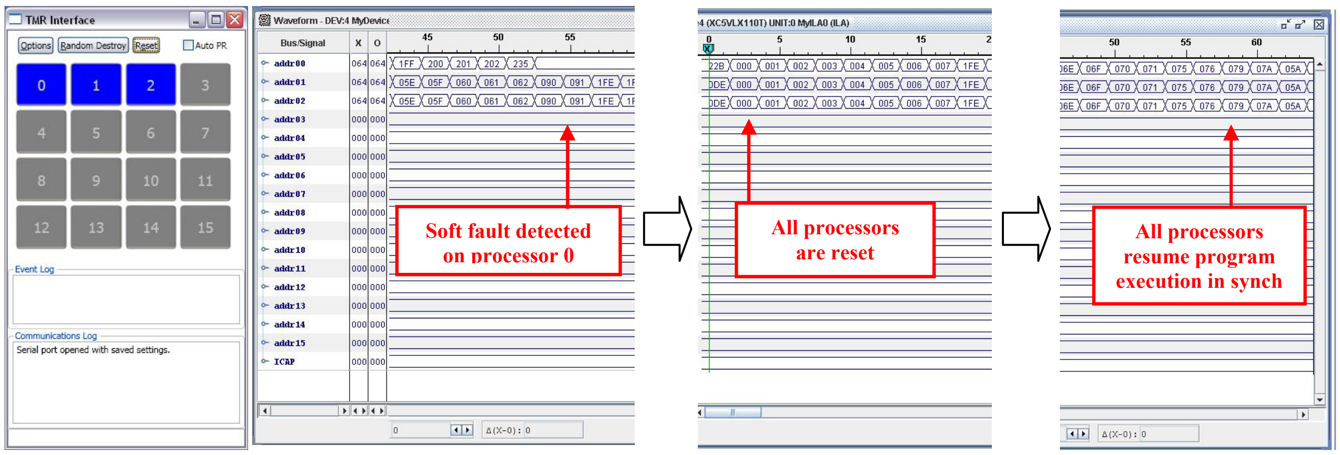
Fig. 5. Flow chart of many-core system operation.

The *Xilinx ChipScope Internal Logic Analyzer* was used to observe the address and data lines between the processors and their instruction memory. Figure 6 shows the *ChipScope* view of system operation after a reset with processors 0, 1, and 2 active as indicated in the GUI. Figure 7 shows the *ChipScope* view of the bus signals during a soft fault in the FPGA fabric. The system continues to service the peripherals despite one of the processors being out of synch using the TMR voter circuit. Once the peripherals are serviced, the recovery sequence is initiated on all processors which resets, reinitializes, and resynchronizes all three processors in the TMR triplet.


TMR Interface
Options
Random Destroy
Reset
Auto PR
Event Log
Communications Log
Waveform - DEV:4 MyDevice4 (XC5VLX110T) UNIT:0 MyILA0 (ILA)
Bus/Signal
Processors 1, 2, and 3 are active and in synch after reset. Processors 3-15 are spares.

Serial port opened with saved settings.

Fig. 6. ChipScope measurement of the address lines on the soft processor after reset showing that processor 0, 1, and 2 are active and in synch (right). Also shown is the corresponding GUI (left) indicating that processors 0, 1, and 2 are active (blue) and the remaining processors are available as spares (gray).


TMR Interface
Options
Random Destroy
Reset
Auto PR
Event Log
Communications Log
Waveform - DEV:4 MyDevice
Bus/Signal
Soft fault detected on processor 0
All processors are reset
All processors resume program execution in synch

Serial port opened with saved settings.

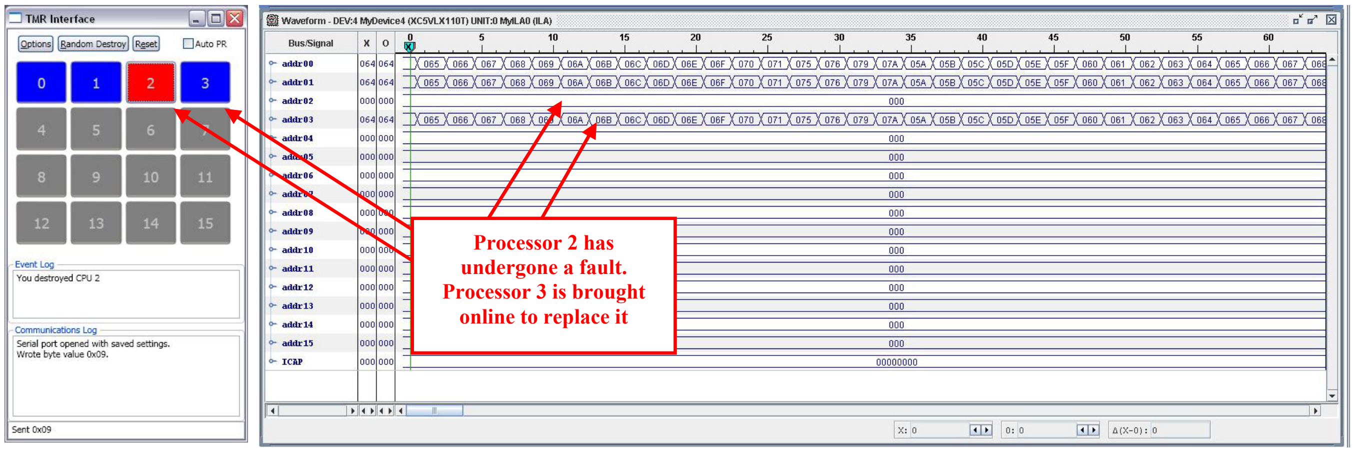
Fig. 7. ChipScope measurement of the address lines on the soft processor showing a soft fault occurring on processor 0. The recovery sequence detects the fault, off loads the good variable data, and then resets and reinitializes all processors.

5. Spatial Avoidance of TID Failures

When an error is detected in the system that is not due to a soft fault in the FPGA fabric, our system attempts to spatially avoid the fault by bringing a new tile online. For our prototype, this type of fault is injected using the GUI or by monitoring back-to-back faults on a processor that has undergone the soft fault recovery sequence (described in previous section). Spatial avoidance of is applicable for tiles that have undergone functional failures due to TID.

The process for enabling a new tile is identical to the process flow chart in figure 5. When the TMR voter detects an error that cannot be recovered using the soft recovery process, it initiates an interrupt to all processors indicating that a fault has occurred. The interrupt service routine sets the *Error_Flag* in the processors. After the processors complete servicing the peripherals in the main program loop, they proceed to offload their variable data to the recovery state machine. The recovery circuit then resets all processors. Upon reset, the recovery circuit selects a spare tile to replace the faulted processor. As the new processor comes out of reset, it is initialized with the same variable data that the two remaining good processors are loaded with. In this way, any spare processor can be brought online and synchronized with two other processors in order to form a TMR triplet. The system contains a log of which processors are available as spares and which ones are marked as *damaged*.

Figure 8 shows the ChipScope measurement of a system which has undergone a fault on processor 2. After an unsuccessful soft fault recovery (section 4), the system brings processor 3 online to form the TMR triplet. The new processor is reset at the same time as the two remaining good processors and all three are loaded with the same variable data. Figure 9 shows the ChipScope measurement for multiple faults in the system. In this figure, processors 2, 4, 6, and 7 have been faulted. The system has activated processors 3, 5, and 8 to form the TMR triplet. This recovery process will continue until only 3 functioning tiles remain.

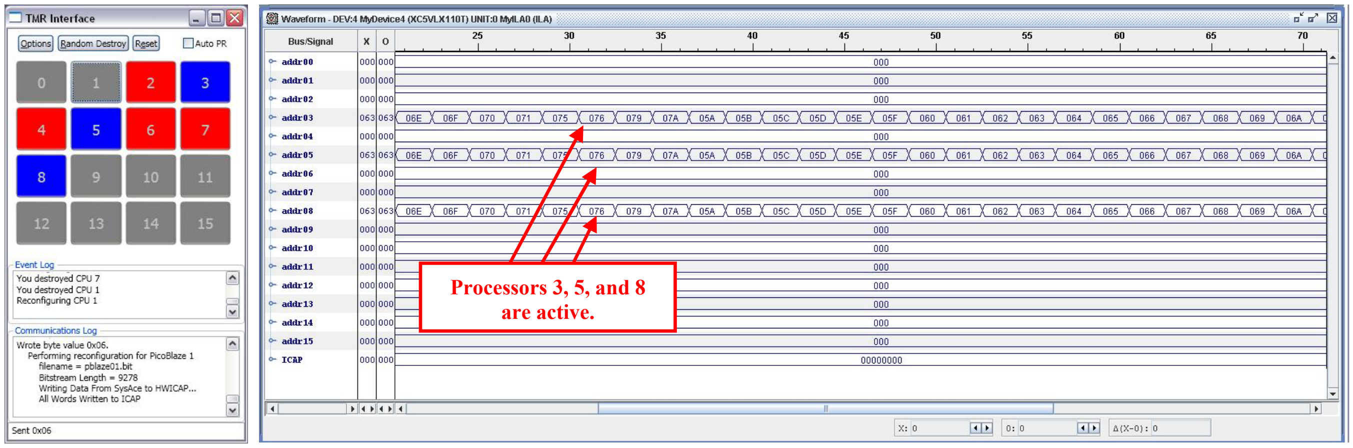

TMR Interface
Options
Random Destroy
Reset
Auto PR
Event Log
Communications Log
Waveform - DEV:4 MyDevice4 (XC5VLX110T) UNIT:0 MyILA0 (ILA)
Bus/Signal
Processor 2 has undergone a fault. Processor 3 is brought online to replace it

```
You destroyed CPU 2
```

```
Serial port opened with saved settings.
Wrote byte value 0x09.
```

```
Sent 0x09
```

Fig. 8. ChipScope measurement of the address lines on the soft processor showing spatial avoidance of faults. In this figure, processor 2 has undergone a fault as indicated by the red in the GUI (left). The system brings processor 3 online to form the TMR triplet and continues operation. Processors 4 through 15 are still available as spares.


TMR Interface
Options
Random Destroy
Reset
Auto PR
Event Log
Communications Log
Waveform - DEV:4 MyDevice4 (XC5VLX110T) UNIT:0 MyILA0 (ILA)
Bus/Signal
Processors 3, 5, and 8 are active.

```
You destroyed CPU 7
You destroyed CPU 1
Reconfiguring CPU 1
```

```
Wrote byte value 0x06.
    Performing reconfiguration for PicoBlaze 1
        filename = pblaze01.bit
        Bitstream Length = 9278
        Writing Data From SysAce to HWICAP...
        All Words Written to ICAP
```

```
Sent 0x06
```

Fig. 9. ChipScope measurement of the multiple processor faults. Processors 2, 4, 6,and 7 have faulted as indicated in the GUI (red). The system spatially avoids these tiles by activating processors 3,5 and 8.

6. SEFI Recovery Using PR

Section 5 described how this system spatially avoids faulted tiles that have undergone a failure that cannot be repaired using the soft fault recovery sequence. Once the system has brought a new tile online, a repair attempt is made on the faulted tile using partial reconfiguration. This type of recovery process will mitigate SEUs that have occurred in the reconfiguration SRAM of the FPGA (i.e., SEFIs). The recovery sequence is performed independently of the normal system operation. Once the tile has been reconfigured, it is entered back into the system's log as an available spare.

Tile Sizing

When designing a system to exploit partial reconfiguration, the first step is to select a tile size that has two features. First, the tile must be sized such that it contains the smallest quantum of resources that will implement the circuit to be reconfigured. In this prototype, the smallest circuit block that was to be reprogrammed was a *picoBlaze* soft processor. The second feature of the PR tile is that it is as small as possible while still meeting the requirements of the partial reconfiguration capability of the FPGA.

For this project, a *picoBlaze* processor was found to require 24 CLBs and 4 BRAMs. The Xilinx PR tools allow PR tiles to be reconfigured in groups of 20 CLBs at a time. Furthermore, the PR tool require that 4 BRAMs be reconfigured at a single time. Due to these requirements, the size of the smallest PR tile for this system is **40 CLBs and 4 BRAM.** The limitation of the PR tool leads to some inefficiencies in the system due to unused resources within the tile. For this project, each tile contains 16 CLBs and 3 BRAMs that are unused but must be included in the partial reconfiguration due to the requirements of the PR guidelines. For perspective, the V5-LX110 FPGA has sufficient resources to implement over 100 picoBlaze processors; however, due to the limitations of the PR tile size, only a 16-core system was able to be implemented with PR. Figure 10 shows a zoom of the floor plan for one of the PR tiles used in this work. Highlighted in this figure are the CLB columns (20 each) and the BRAM columns (4 each).

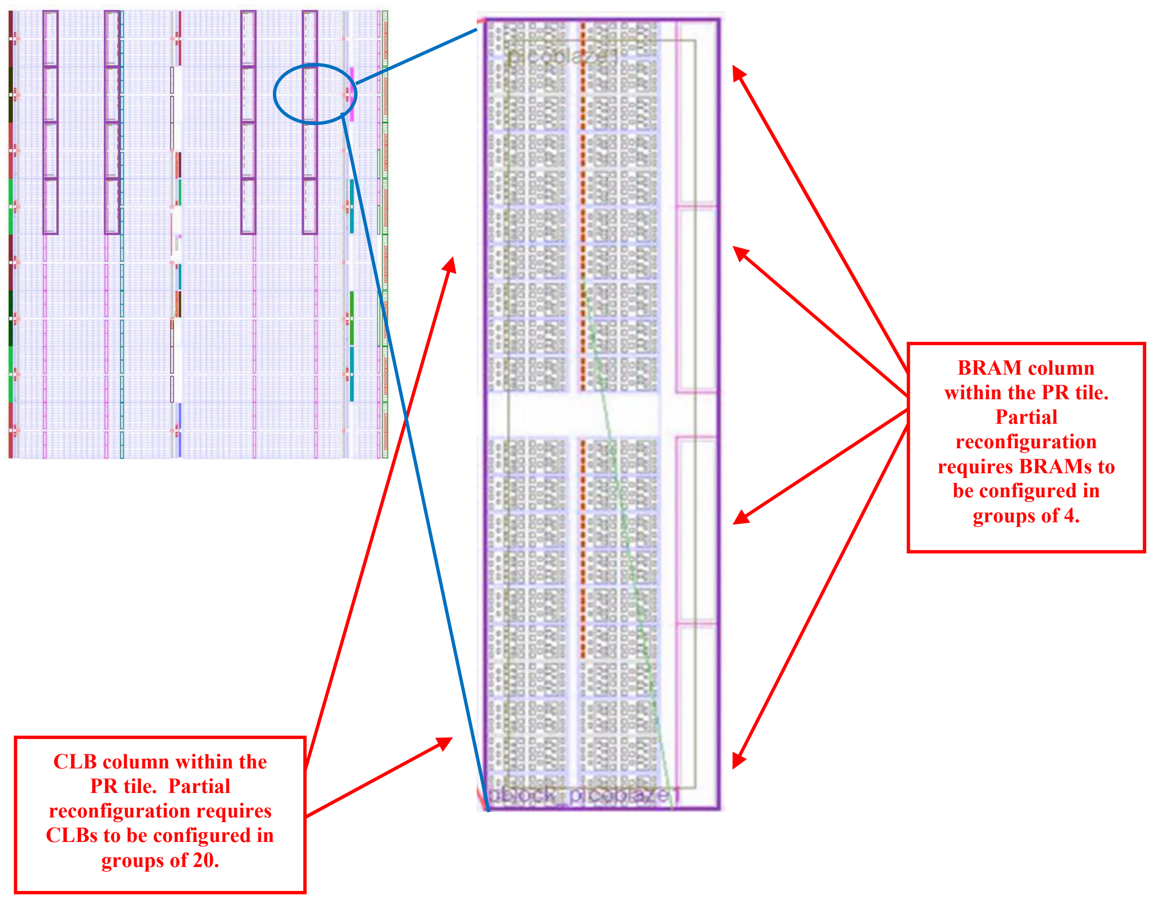

picoblaze1
block picoblaze1
BRAM column within the PR tile. Partial reconfiguration requires BRAMs to be configured in groups of 4.
CLB column within the PR tile. Partial reconfiguration requires CLBs to be configured in groups of 20.

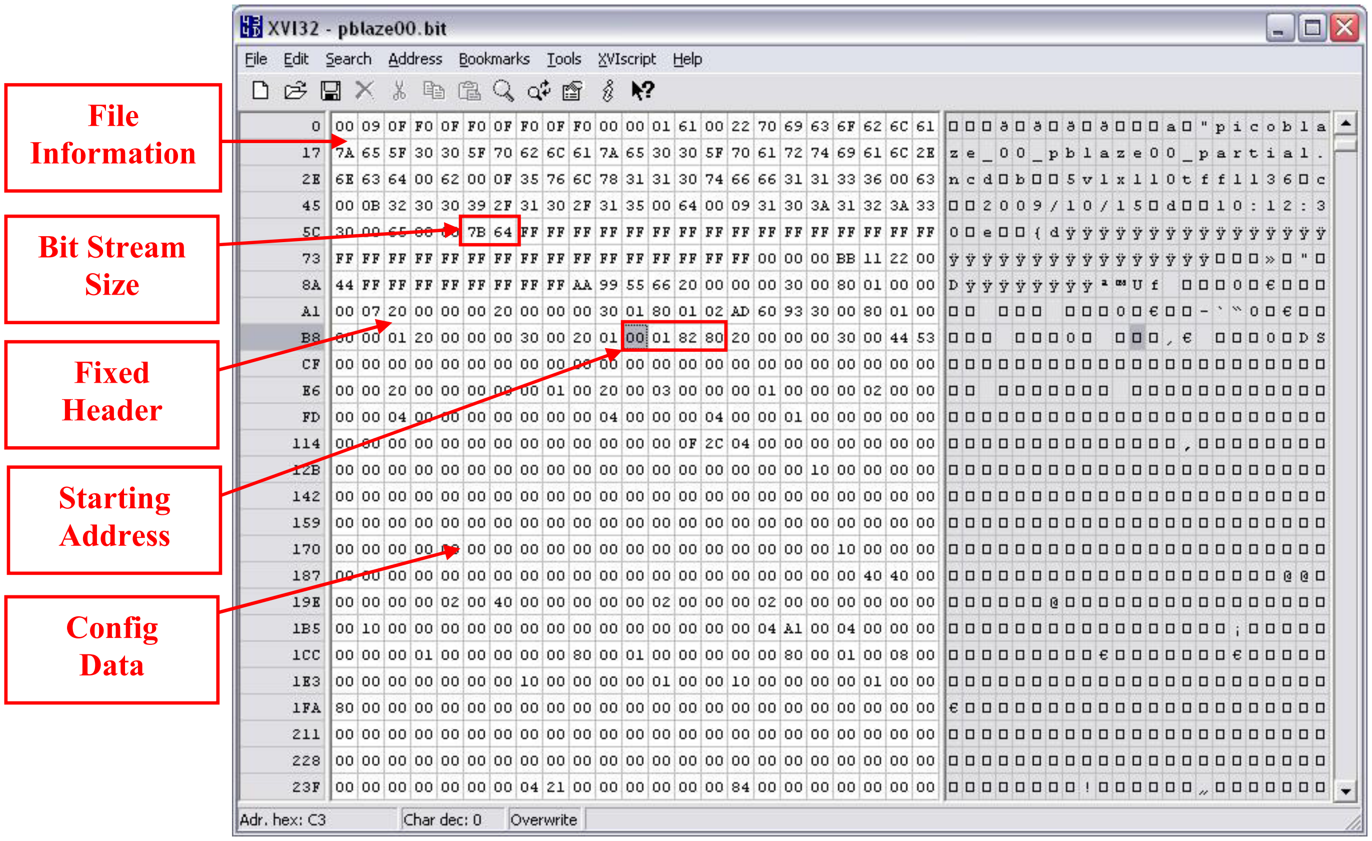
Fig. 10. Zoomed in view of the PR tile used in this work highlighting the constraints of partial reconfiguration.

Partial Reconfiguration Bit Streams

Each of the 16 tiles in this system contains a unique configuration bit stream file (32k byte). These bit streams were generated after the floor planning of the tiles and contain the specific addresses and configuration bits for their corresponding tile. These configuration files are stored off-chip in a non-volatile Flash EEprom. The bit file is retrieved from non-volatile memory using the Xilinx SystemACE component. Each of the bit stream files contains a header with unique information about the bit stream file. This is then followed by information regarding the size of the bit stream file. This is then followed by a constant header of 54 bytes. After this, unique configuration data is sent starting with a unique starting address corresponding to a location in the configuration SRAM. Table 1 shows the starting address for each of the partial reconfiguration bit files in this design. This information is useful when monitoring the partial reconfiguration in real-time using ChipScope. Figure 11 shows the HEX and ASCII translation of the bit stream file for Tile 0 in our system with the important words highlighted.

picoBlaze Tile	Configuration SRAM Starting Address (HEX)
0	x00018280
1	x00018780
2	x00019400
3	x00019980
4	x00010280
5	x00010800
6	x00011400
7	x00011980
8	x00008280
9	x00008800
10	x00009400
11	x00009980
12	x00000280
13	x00000800
14	x00001400
15	x00001980

TABLE I
TILE PR STARTING ADDRESSES FOR CONFIGURATION SRAM



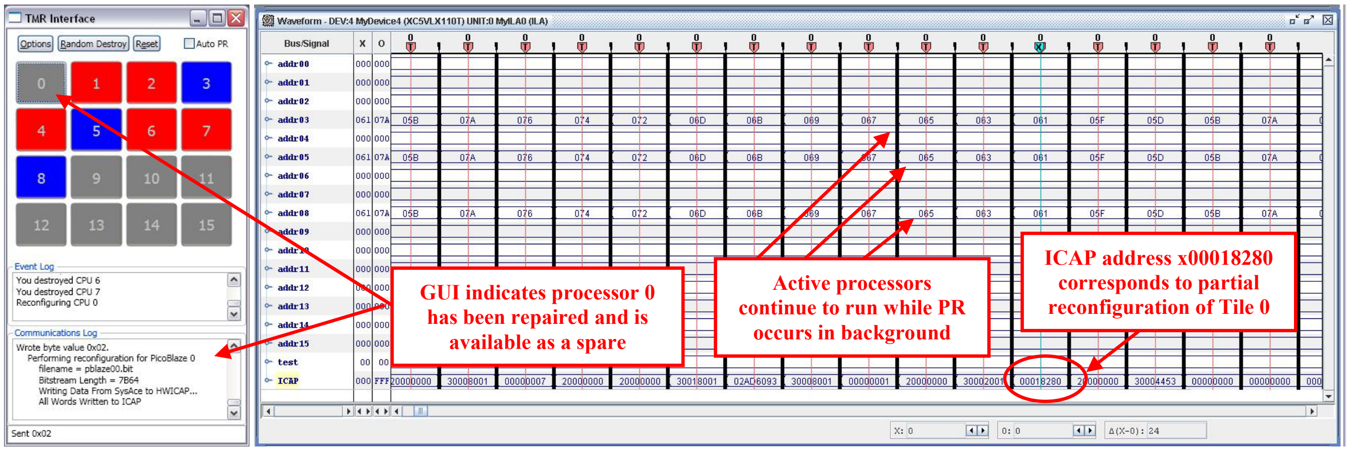
```
   0  00 09 0F F0 0F F0 0F F0 0F F0 00 00 01 61 00 22 70 69 63 6F 62 6C 61
  17  7A 65 5F 30 30 5F 70 62 6C 61 7A 65 30 30 5F 70 61 72 74 69 61 6C 2E
  2E  6E 63 64 00 62 00 0F 35 76 6C 78 31 31 30 74 66 66 31 31 33 36 00 63
  45  00 0B 32 30 30 39 2F 31 30 2F 31 35 00 64 00 09 31 30 3A 31 32 3A 33
  5C  30 00 65 00 00 7B 64 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
  73  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 BB 11 22 00
  8A  44 FF FF FF FF FF FF FF FF AA 99 55 66 20 00 00 00 30 00 80 01 00 00
  A1  00 07 20 00 00 00 20 00 00 00 30 01 80 01 02 AD 60 93 30 00 80 01 00
  B8  00 00 01 20 00 00 00 30 00 20 01 00 01 82 80 20 00 00 00 30 00 44 53
  CF  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  E6  00 00 20 00 00 00 00 00 01 00 20 00 03 00 00 00 01 00 00 00 02 00 00
  FD  00 00 04 00 00 00 00 00 00 00 04 00 00 00 04 00 00 01 00 00 00 00 00
 114  00 00 00 00 00 00 00 00 00 00 00 00 00 0F 2C 04 00 00 00 00 00 00 00
 12B  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00
 142  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 159  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00
 187  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 40 00
 19E  00 00 00 00 02 00 40 00 00 00 00 00 02 00 00 00 02 00 00 00 00 00 00
 1B5  00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 A1 00 04 00 00 00
 1CC  00 00 00 01 00 00 00 00 00 80 00 01 00 00 00 00 00 80 00 01 00 08 00
 1E3  00 00 00 00 00 00 00 10 00 00 00 00 01 00 00 10 00 00 00 00 01 00 00
 1FA  80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 211  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 228  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 23F  00 00 00 00 00 00 00 04 21 00 00 00 00 00 00 84 00 00 00 00 00 00 00

Adr. hex: C3    Char dec: 0    Overwrite
```

Fig. 11. Partial reconfiguration bit stream file contents for tile 0 of our system highlighting key portions of file.

A dedicated *MicroBlaze* soft processor is used to handle the independent partial reconfiguration of the tiles. The *MicroBlaze* communicates directly with the *Xilinx SystemACE* component to retrieve the individual PR bit streams. The *SystemACE* communicates with the off-chip Flash EEprom (figure 3) and reads the corresponding bit files out of non-volatile memory. The *MicroBlaze* processor takes the information from the *SystemACE* and passes it into the *HWICAP* core. This core is generated using the Xilinx PR tools. Within this core are a series of FIFO blocks which handle retiming the bit stream files so that they can be driven into the ICAP port. The ICAP port provides direct access to the configuration SRAM and enables partial reconfiguration.

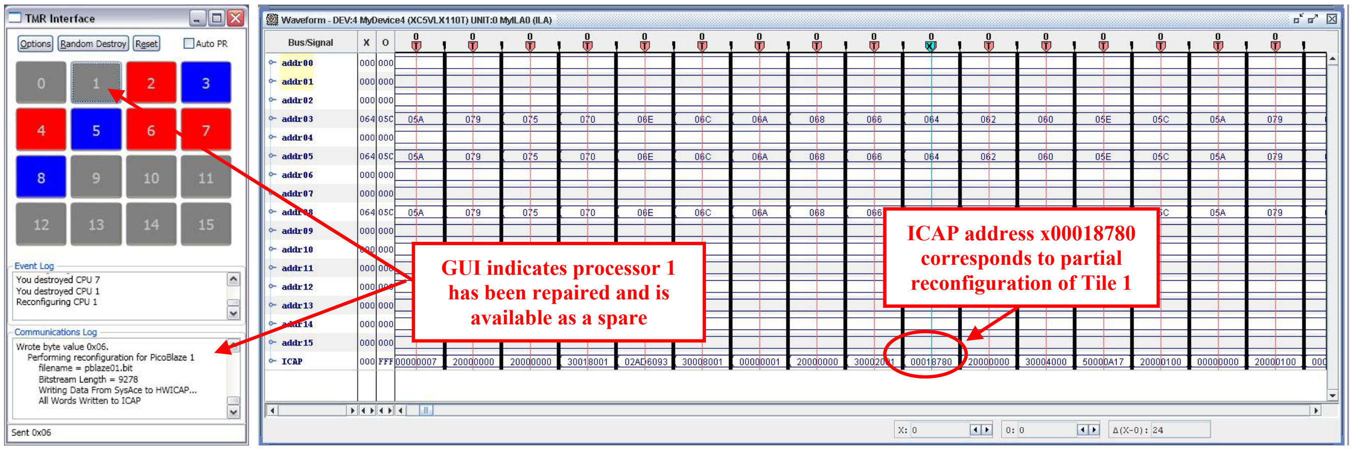
In order to monitor the partial reconfiguration occurring in the background while the processors are running, the ICAP port signals are observed with ChipScope. Figure 12 shows the system during the reconfiguration of tile 0. In this figure, ChipScope displays the address busses of the three active processors (3, 5 and 8). In this case, processors 0, 1, 2, 4, 6, and 7 are faulted with a SEFI. The system is currently repairing processor 0 by reprogramming its tile. The GUI indicates that processor 0 is being repaired and will be available as a spare upon completion by turning it from red to gray. ChipScope is displaying the signals that are being driven into the ICAP port. As the ICAP data is monitored, the unique starting address for tile 0 (x00018280) appears on the bus indicating that tile 0 is being reprogrammed in the background as processors 3, 5, and 8 continue to run.

Figure 13 shows the recovery of a SEFI on processor 1 through partial reconfiguration of tile 1. The GUI indicates that processor 1 is being repaired and will be available as a spare upon completion by turning it from red to gray. Again, ChipScope monitors the ICAP data and observes the unique starting address for tile 1 (x00018780) on the bus indicating that tile 1 is being reprogrammed in the background as processors 3, 5, and 8 continue to run.


TMR Interface
Options
Random Destroy
Reset
Auto PR
Waveform - DEV:4 MyDevice4 (XC5VLX110T) UNIT:0 MyILA0 (ILA)
GUI indicates processor 0 has been repaired and is available as a spare
Active processors continue to run while PR occurs in background
ICAP address x00018280 corresponds to partial reconfiguration of Tile 0

```
Event Log
You destroyed CPU 6
You destroyed CPU 7
Reconfiguring CPU 0

Communications Log
Wrote byte value 0x02.
    Performing reconfiguration for PicoBlaze 0
        filename = pblaze00.bit
        Bitstream Length = 7B64
        Writing Data From SysAce to HWICAP...
        All Words Written to ICAP

Sent 0x02
```

*Fig. 12. Recovery from SEFI on **processor 0** using partial reconfiguration. The ICAP port is monitored using ChipScope showing the partial reconfiguration of tile 0 (address x00018280) while processors 3, 5, and 8 are active.*


TMR Interface
Options
Random Destroy
Reset
Auto PR
Waveform - DEV:4 MyDevice4 (XC5VLX110T) UNIT:0 MyILA0 (ILA)
GUI indicates processor 1 has been repaired and is available as a spare
ICAP address x00018780 corresponds to partial reconfiguration of Tile 1

```
Event Log
You destroyed CPU 7
You destroyed CPU 1
Reconfiguring CPU 1

Communications Log
Wrote byte value 0x06.
    Performing reconfiguration for PicoBlaze 1
        filename = pblaze01.bit
        Bitstream Length = 9278
        Writing Data From SysAce to HWICAP...
        All Words Written to ICAP

Sent 0x06
```

*Fig. 13. Recovery from SEFI on **processor 1** using partial reconfiguration. The ICAP port is monitored using ChipScope showing the partial reconfiguration of tile 0 (address x00018780) while processors 3, 5, and 8 are active.*

7. Parametric Performance

Soft Fault Recovery

The amount of time it takes to perform a soft fault recovery depends on the number of clock cycles it takes to generate the *Interrupt* and *Reset* plus the time it takes to off-load the variable information from a good processor and then re-load the variable information back into the three active processors. It takes two clocks for both the interrupt and reset in our system. Each *picoBlaze* processor contains 64 bytes of RAM data that needs to be off-loaded and reloaded during each recovery. Each read/write takes 2 clocks per bus cycle. Our system runs off of a 100MHz system clock. The timing overhead to perform a soft fault recovery was found to be **2.6us** using the following:

$$t_{SoftFault\,Recovery} = t_{IRQ} + t_{RST} + t_{Var\_Out} + t_{Var\_In}$$

$$t_{SoftFault\,Recovery} = 2clks + 2clks + 128clks + 128clks$$

$$t_{SoftFault\,Recovery} = 260clks = \left({}^{1}\!/_{100MHz} \right) = 2.6us$$

Spatial Avoidance of TID

The amount of time to bring a spare processor online is identical to the time it takes to perform a soft fault recovery procedure (**2.6us**). The only difference in the procedures is that when the variable data is loaded into the three processors, one of the active processors was previously a spare.

SEFI Recovery

The amount of time that it takes to perform partial reconfiguration of a tile depends on the size of the tile and the speed at which data can be written to the ICAP port. For our system, the time for PR was obtained empirically by measuring the time between the start and stop of the reconfiguration. The PR for each tile took approximately 200 clocks for each byte of configuration data. Our tiles were 31.2k bytes in size, which corresponded to a tile PR time of **66ms**. For comparison, an entire V5-LX110 device requires a bit stream size of 3,799k bytes to reconfigure and takes 984ms using the SystemACE.

8. Conclusion

This paper presented the design and prototyping of a many-core computer architecture with fault detection and recovery. Our approach uses three fault mitigation techniques to recovery from radiation induced failures on an FPGA. The system partitions an FPGA into equally sized tiles, each containing a soft processor. An SEU in the FPGA fabric is mitigated using a reset sequence. An SEU in the configuration SRAM (ie., a SEFI) is mitigated using partial reconfiguration of the tile. Finally, TID damage in a tile is mitigated using spatial avoidance of the effected region. The soft fault recovery and spatial TID avoidance strategies were found to take 2.6us to complete. The partial reconfiguration of a single tile was found to take 66ms to complete. This type of comprehensive fault mitigation strategy addresses the three main failure mechanisms in FPGA-based computing systems (SEU in the fabric, SEFIs, and TID damage) and can be used to improve the reliability of FPGA-based flight computers for military and aerospace applications.

Acknowledgments

The authors would like to thank the Montana Space Grant Consortium and the NASA *Explorations Systems Mission Directorate Higher Education Project* for support of this work. The authors would also like to thanks Dr. Robert E. Ray and Dr. Andrew S. Keys of the NASA Advanced Avionics and Processor Systems (AAPS) project for their mentoring of this research.

References

[1] A. Holmes-Siedle, L. Adams, "Handbook of radiation Effects", 2nd edition, New York, Oxford University Press, 2002.

[2] C. Claeys, E. Simoen, "Radiation Effects in Advanced Semiconductor Materials and Devices", Berlin Heidelberg, ISBN 3-540-43393-7, Springer-Verlag, 2002.

[3] John Cochran – "A SET Resistant Majority Voting Circuit", Military / Aerospace Programmable Logic Devices (MAPLD) Conference, 2009.

[4] Melanie Berg, "Design for Radiation Effects", Military / Aerospace Programmable Logic Devices (MAPLD), Conference, Annapolis, MD, 2008.

[5] M. Stetller, "Radiation Effects and Mitigation Strategies for modern FPGAs", 10th annual workshop for LHC and Future experiments, Los Alamos National Laboratory, USA, 2004.

[6] Melanie Berg, "Embedding Asynchronous FIFO Memory Blocks in Xilinx Virtex Series FPGAs Targeted for Critical Space System Applications", Military/Aerospace PLD (MAPLD) Conference, 2009.

[7] C. Carmichael, B. Bridgford, G. Swift, M. Napier, "A Triple Module Redundancy Scheme for SEU Mitigation of Static Latch-Based FPGAs", Military/Aerospace PLD (MAPLD) Conference, 2004.

[8] Ricky W. Butler, "A Primer on Architectural Level Fault Tolerance", NASA Scientific and Technical Information (STI) Program Office, Report No. NASA/TM-2008-215108, Feb. 2008.

[9] Mehlitz, P.C., Penix, J.J., and Markosian, L. Z., "Radiation-Hardened Software for Space Flight Science Applications", American Geophysical Union, Fall Meeting 2005, abstract #IN41A-0316, 2005.

[10] S. Franklin, K. Jentung, B. Spence, M. McEachen, S. White, J. Samson, R. Some, J. Zsoldos, "The Space Technology 8 Mission", 2006 IEEE *Aerospace Conference*, pp 16, March 4-11, 2006.

[11] J. Greco, G. Cieslewski, A. Jacobs, I.A. Troxel, A.D. George, "HW/SWinterface for high-performance space computing with FPGA coprocessors", 2006 IEEE *Aerospace Conference*, pp 10, March 4-11, 2006.

[12] G. Alonzo Vera – "A Programmable Configuration Scrubber for FPGAs", Military / Aerospace Programmable Logic Devices (MAPLD) Conference, 2009.

[13] A. Keys, J. Adams, R. Darty, M. Patrick, M. Johnson, & J. Cressier "Radiation Hardened Electronics for Space Environments (RHESE) Project Overview ", International Planetary Probes Workshop (IPPW), Atlanta, GA, June 2008.

[14] J.W. Gambles, G.K. Maki, "Rad-Tolerant Flight VLSI From Commercial Foundries", *39th IEEE Midwest Symposium on Circuits and Systems*, vol 3, pp. 1227-1230, August 18-21, 1996.

[15] M. Johnson, "Radiation Hardened, High Performance, Power Efficient Processing – An Objective of the NASA Exploration Systems Technology Development Program", *13th NASA Symposium on VLSI Design*, Post Falls, ID, June 5-6, 2007.

[16] Bernard Bancelin – "ATMEL ATF280E Rad Hard SRAM Based FPGA: SEE test results and fault injection", Military / Aerospace Programmable Logic Devices (MAPLD) Conference, 2009.

[17] Andrew S. Keys, Howell, "Technology Developments in Radiation-Hardened Electronics for Space Environments", NASA Technical Reports Server (NTRS), Document No. 20080032798, [Online], Available: ntrs.nasa.gov, June. 2008.

[18] Marshall C. Patrick, "RHESE Reconfigurable Computing (RC) Task", Military / Aerospace Programmable Logic Devices (MAPLD), Conference, Annapolis, MD, 2008.

[19] "PicoBlaze™ 8-bit Embedded Microcontroller User Guide", Xilinx Document No. UG129 (v1.1.2), [Online], Available: www.xilinx.com, June. 2008.

Biography

Clint Gauer *(M'06) received the B.S. degree in computer engineering from Montana State Univ., Bozeman in 2008 and is currently an M.S. degree candidate in electrical engineering at Montana State Univ., Bozeman with an expected graduation date of May 2010.*

He is currently a Research Assistant in the Department of Electrical and Computer Engineering at Montana State University (MSU), Bozeman where his focus is on reconfigurable computing architectures for mitigating system level faults due to space radiation.

Brock J. LaMeres *(M'98-SM'09) received the B.S. degree in electrical engineering from Montana State Univ., Bozeman in 1998, and the M.S. degree in electrical engineering from the Univ. of Colorado, Colorado Springs in 2001, and the Ph.D. degree in electrical engineering from the Univ. of Colorado, Boulder in 2005.*

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Montana State University (MSU), Bozeman. LaMeres teaches and conducts research in the area of digital systems. Prior to joining the faculty at MSU in 2006, he worked as a Hardware Design Engineer for Agilent Technologies in Colorado Springs from 1999 to 2006. LaMeres' research is sponsored by NASA, the National Science Foundation, the Montana Space Grant Consortium, the National Space Grant Consortium, and the Office of Naval Research.

David Racek *(M'06) is currently an B.S. degree candidate in computer engineering at Montana State Univ., Bozeman with an expected graduation date of Dec. 2009.*

He is currently a Research Assistant in the Department of Electrical and Computer Engineering at Montana State University (MSU), Bozeman where his focus is on exploiting many-core computer architectures for increased fault immunity. Racek has held research positions at the Jet Propulsions Laboratory and at the MSU Space Science and Engineering Laboratory (SSEL) where he has worked on flight computer systems for small satellites.