Single Event Effects (SEE) Mitigation of Reconfigurable FPGAs

David R. Czajkowski, Manish P. Pagey, Murat Goksel, David J. Bozek
Space Micro Inc.
10401 Roselle Street STE 400, San Diego, CA 92121; 858-332-0700
dcz@spacemico.com, pagey@spacemicro.com, dbozek@spacemicro.com

ABSTRACT: This paper discusses the application of Space Micro's Time-Triple Modular Redundancy (TTMR™) and Hardened-Core (H-Core™) technologies for mitigation of Single Event Effects in Xilinx Virtex-II field programmable gate arrays (FPGA). TTMR is the application of time and spatial redundancy for mitigating Single Event Upsets (SEU). H-Core is an auxiliary rad-hard chip used for mitigating Single Event Functional Interrupts (SEFI). These technologies have been proven to perform in proton radiation environments and are presently the driving force behind our powerful rad-hard computers built from COTS processors.

These same technologies with a few modifications have been applied to mitigate SEUs, configuration upsets, and SEFIs in Virtex-II FPGAs. The application of TTMR and H-Core2 has been successfully demonstrated to mitigate SEUs and SEFIs under radiation. The target system has been named the Proton300k™. The Proton300k is an ideal reconfigurable computing platform for radiation hardened, high performance data processing at low cost and outperforms similar systems built from expensive rad-hard electronics (from rad-hard processes).

1. INTRODUCTION

Reprogrammable FPGAs provide the system designer with several advantages compared to those FPGAs that are one-time programmable (OTP). One particular advantage is due to their use of multiple programmable memory cells (usually SRAM or Flash) that allow their configuration to be changed not only during the design phase, but also after installation into an operational system if needed.

This ability to be reprogrammed can however have detrimental effects when it comes to using these devices for aerospace applications. Single Event Upsets (SEU) (or bitflips) may cause the data stored in these memory cells to corrupt. Such errors may corrupt data, lead to mission critical configuration errors, or even hang or reset the FPGA (the latter known as Single Event Functional Interrupts or SEFIs).

There are several known techniques that can help to mitigate SEU effects in SRAM-based Xilinx Virtex FPGAs (which are the focus of this research), Readback, Reconfiguration (full and partial), Scrubbing, and Modular Redundancy, when used individually or in combination can be used to mitigate SEFIs and SEUs.

The published methods for eliminating SEUs in FPGAs can result in great increases in the cost of the design. The cost growth can become so prohibitive that it can come close to the cost of a system designed using parts which are based upon a radiation hardened process (a process which is inherently tolerant to radiation effects).

One of the widely used methods to build SEU hardened systems based on Xilinx FPGAs is to employ Triple Modular Redundancy (TMR). This approach to designing an SEU tolerant system comes at the expense of excessive area, power, and cost overhead. In addition, using a TMR approach with Virtex FPGAs increases the required I/O pin-count to 3 times the original. This can a limiting factor on the size of the design that can be mapped on the FPGA, and also places constraints upon how the FPGAs may be incorporated onto a circuit board.

TMR for FPGAs requires that the voter itself be rad-hard, and normally this is implemented by using a rad-hard application specific integrated circuit (ASIC). This adds even more complexity and additional power requirements to the circuitry. In addition, the use of an ASIC reduces the overall reconfigurability of the fielded system.

There has been some debate as of late as to whether TMR is warranted for all missions [5]. Design trends have shifted away from a focus purely upon providing a radiation-hardened system, towards also keeping costs down (while providing the same level of hardening and circuit performance). There has been much focus in the

rad hard design community upon developing methods that would lessen (or nullify) the penalties associated with providing rad hard solutions to the use of FPGAs, while maintaining the advantages of FPGA architectures such as those in the Virtex line.

The following sections discuss the proposed methods that aim to aid designers in developing a SEU tolerant system that is less expensive and provides an acceptable level of performance without resorting to rad-hard parts.

2. RADIATION EFFECTS IN XILINX FPGAs

The attractive feature of SRAM based FPGAs is their ability to be reprogrammed. The way this feature is implemented however can create problems under harsh radiation environments.

The reprogrammable SRAM cells used in the Virtex FPGA are highly sensitive to SEUs. Since almost all of the functionality of the FPGA is dependant upon using SRAM cells, more "chip area" is susceptible to SEUs. Therefore, Xilinx FPGAs have more susceptibility and are prone to a higher number of SEUs. An SEU in the cells responsible for storing data corresponding to control logic can lead to a SEFI. These SEFIs result in the interruption of service, which can be quite undesirable in most cases.

The sensitivity of Xilinx FPGAs is shown in the following radiation data[1], Figures 1 and 2. Both of these SEU/SEFI cross-section curves highlight the very low SEU and SEFI thresholds (approximately 1 Mev/mg/cm2), making these devices very sensitive.

In summary, Xilinx based FPGAs suffer from the following 3 major SEE problems:

- Data SEUs: Single event upsets (SEU) in an SRAM block dedicated to data. These SEUs lead to the corruption of the data being processed by the FPGA.
- Configuration SEUs: SEUs in the memory cells storing the configuration data can alter the functionality of the FPGA.
- FPGA SEFI: SEUs occurring in some areas of the FPGA can lead to SEFI. These SEFIs (several types of which are discussed below) lead to improper operation of the device and are mitigated by reconfiguring the device.

SEFIs in Virtex are classified into three different types depending upon in which part of the FPGA the SEFI has occurred.

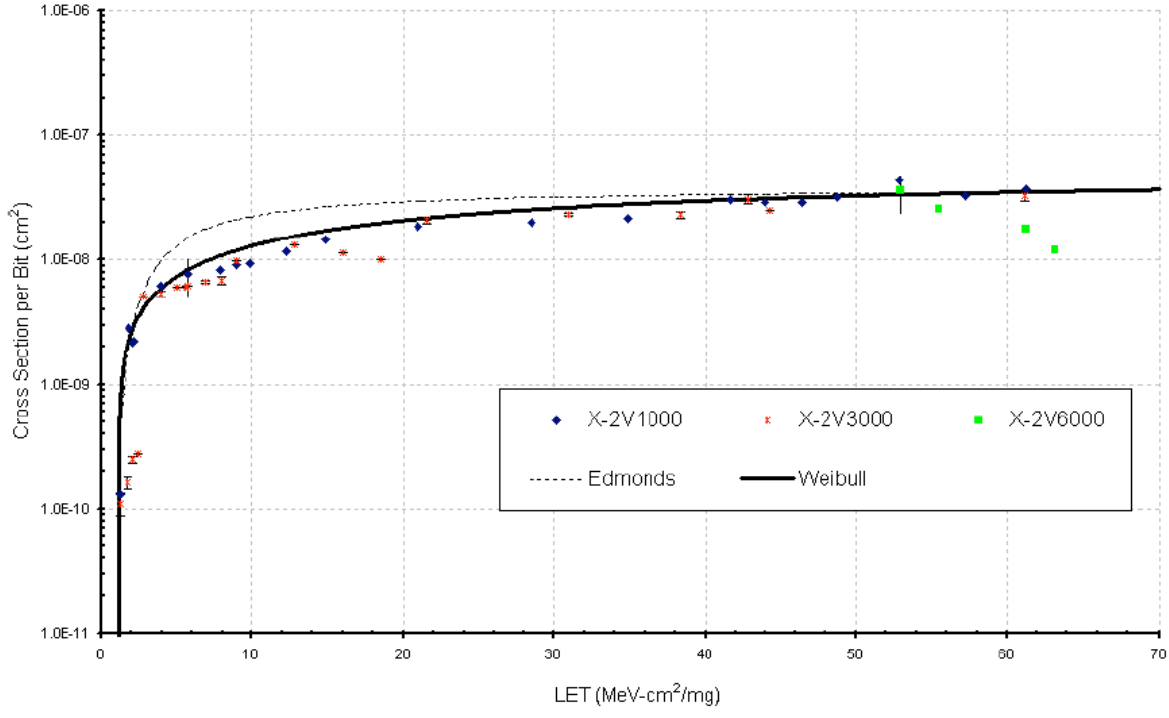

Figure 1. Virtex II configuration memory cells (Texas A&M). Heavy Ions SEU Cross Sections for the X-2V1000, X-2V3000, and X-2V6000

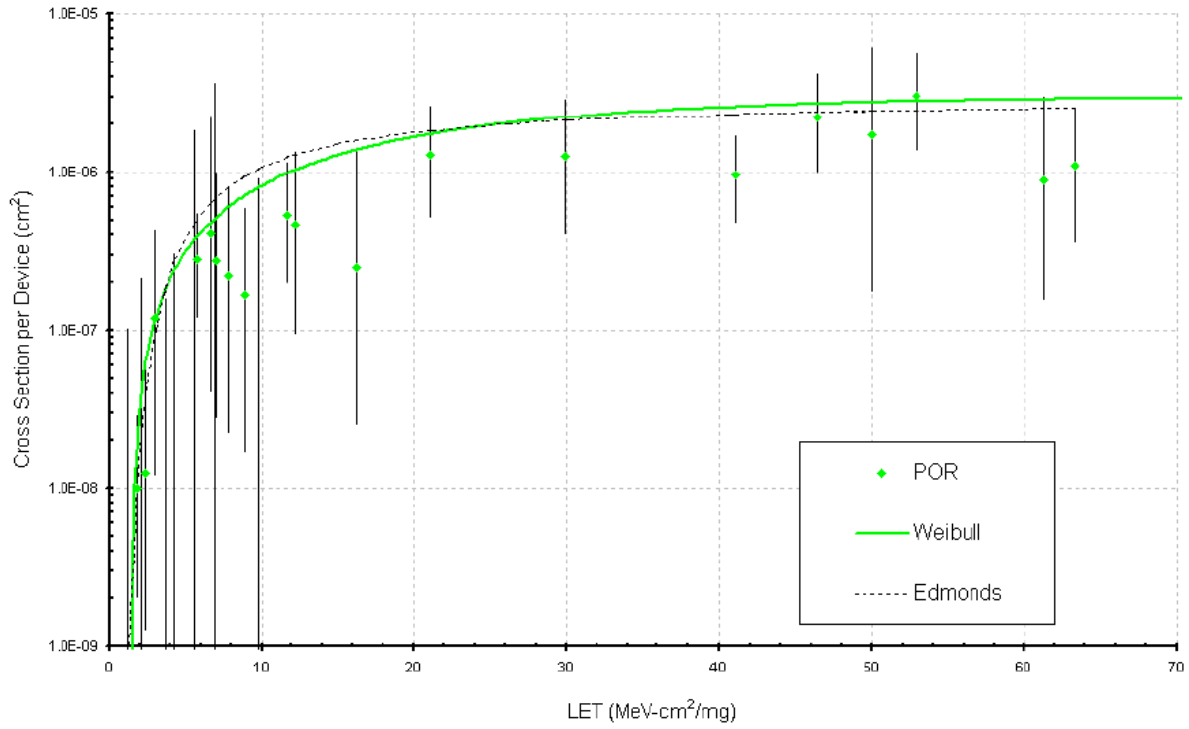

Figure 2. Virtex-II POR SEFI Cross Section

- SEFI in Power on Reset (POR): SEUs are random in nature and can occur in any potentially sensitive part of the device. An SEU in the POR circuitry can lead to spurious transition on the reset line leading to the loss of user and configuration data.

- SEFI in Select Microprocessor Access Port (SMAP): SMAP is a bi-directional interface that can be used to read/write to an FPGA.

- JTAG: A SEFI in JTAG can result in the loss of communication with the configuration logic.

Previous research [13] has shown that reconfiguring the Virtex-II FPGA without powering it down mitigates all of the above SEFI conditions, however, reconfiguring the device results in loss of valuable data.

Xilinx has proposed many different solutions to mitigate the effects of SEUs. These solutions include the addition of redundant logic, use of an external rad-hard ASIC, or/and the use of other features like Readback and Reconfiguration, Scrubbing etc.

3. TTMR AND H-CORE: A QUICK REVIEW

The TTMR and H-Core methods were developed to mitigate SEUs and SEFIs in microprocessors. Below is a brief description of how the two technologies are used with microprocessors. In a later section, a description will show how modified versions of these techniques can be used with reconfigurable FPGAs, such as the Xilinx Virtex family.

TTMR

TTMR is an SEU mitigation method based upon a hybrid of both spatial and time redundancy.

Spatial redundancy is implemented by having multiple copies of logic circuitry placed in parallel, and the same operations are performed in each copy. Since the probability of radiation striking and corrupting more than one of the parallel logic elements is extremely low, the resulting outputs of the multiple logic elements are compared using a radiation hardened "voter" to determine and select the correct result.

Time redundancy is implemented by using a single copy of logic circuitry which then executes a series of operations at multiple, and therefore different, times. The results are stored in a radiation hardened memory location, and then compared using a radiation-hardened voter.

TTMR combines these two techniques and exploits the multiple execution units of a VLIW (Very Long Instruction Word) processor to execute identical

instructions in parallel. To increase the performance of the processor, only two sets of instructions are run first. The results of these two sets are compared and, in the instance of a mismatch, time redundancy is used to run the set of instructions again, and the results are then compared using a voting algorithm.

The flowchart shown in Figure 3 shows the steps that the TTMR algorithm follows. The algorithm starts by loading and running the first instruction (or group of instructions) of the program. “O“ and “M“ represent the “Original“ and “Mirror“ copies of the instruction.

The O and M instructions are spatially separated, by executing them on the different ALUs (arithmetic logic units) of the VLIW microprocessor. Doing this eliminates the points of identical failures that might occur in the two sets as in the case of a microprocessor using time redundancy.

The next step of the flow chart shows that the results of the O and M instructions are then compared to see if they match. If the results match, the current state of the program is saved: the uncorrupted results of the O and M are written back to the main memory. However, a mismatch in the O an M results indicates the occurrence of an SEU.

At this point, another set of results is calculated. Either ALU of the processor, or another ALU, can be used in calculating the results of the “Third” set or “T” instructions. The results of T and O are now compared. If the results match, the M results were corrupted. The O (and T) results are used (the M value is overridden). This process is repeated until the end of the program.

If O and T results do not agree, the T results and M results are compared. If they match, the M results are copied into the O results. The algorithm continues with the execution of other instructions of the program.

At this stage, if a disparity is found in the T and M results, this implies that there was probably more than one SEU during the execution, resulting in an uncorrectable SEU. The probability of such an event is extremely small, because this would indicate that SEUs would have had to occur in separate ALUs within a very small window of time (typically less than 10 clock cycles).

Space Micro successfully demonstrated TTMR using 51 MeV proton radiation at the UC Davis Nuclear Laboratory. TTMR was able to detect and correct SEUs on Texas Instruments 320C6000 DSP, and Equator Technologies BSP-15 processor [16].

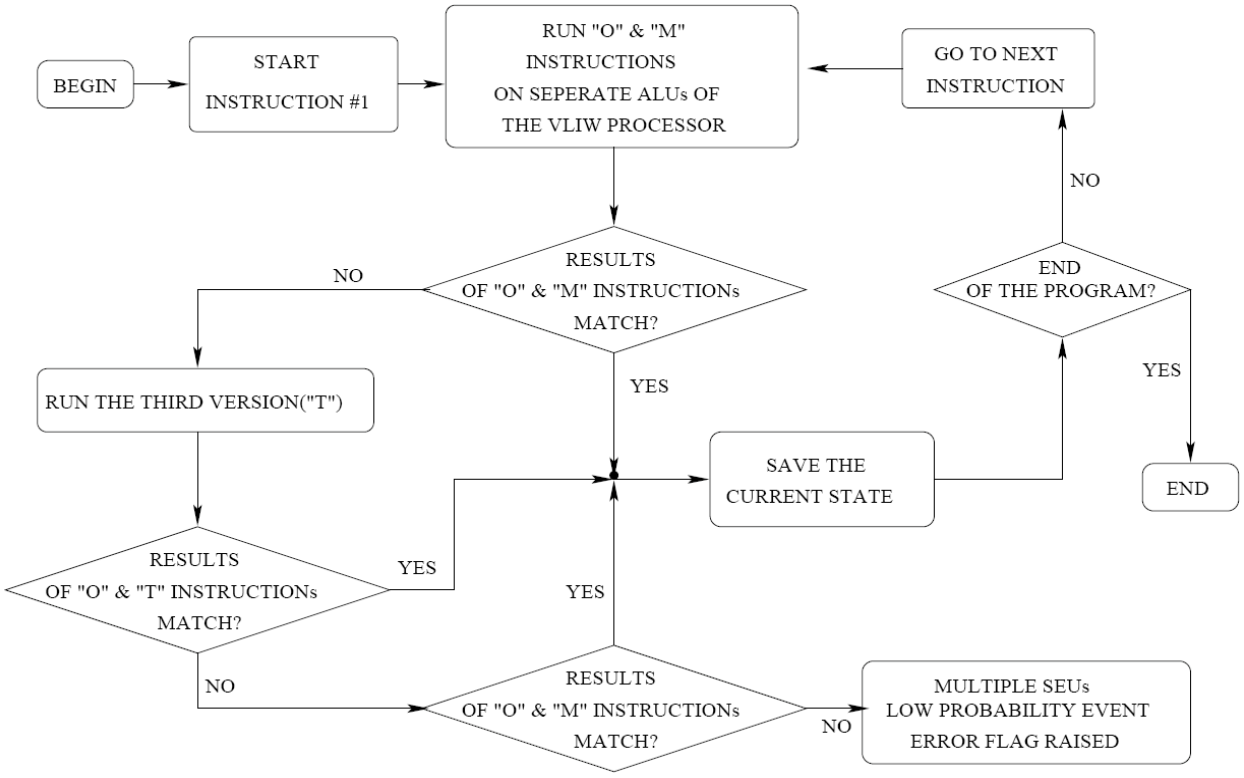

Figure 3: TTMR Algorithm Executing Three Copies of Programs/Instructions

H-Core

SEFI is a Single Event Upset (SEU) in the control logic of electronics that leads them to enter into an unknown state, and ultimately stop responding (hang). This can result in loss of valuable data, or loss of their functionality, or in most cases, both [6].

The previous state-of-the-art method of recovering parts from an SEFI is to power-cycle the device after detecting that the system has ceased to respond. Various methods have been published to detect such a condition.

A serious drawback of this solution is that the system is inoperable for the entire duration from when the SEFI occurs, through detection, until the power is cycled, and the device is restarted. This is undesirable and in some cases unacceptable.

The Hardened Core (H-Core) device is built using a radiation tolerant process (such as Silicon On Insulator, SOI) and can be used for SEFI detection and mitigation. H-Core acts as an external device (relative to the target device to be monitored for SEFI events) that monitors the target and is able to sense when an SEFI has occurred. A photo of Space Micro's H-Core chip is shown in Figure 4.

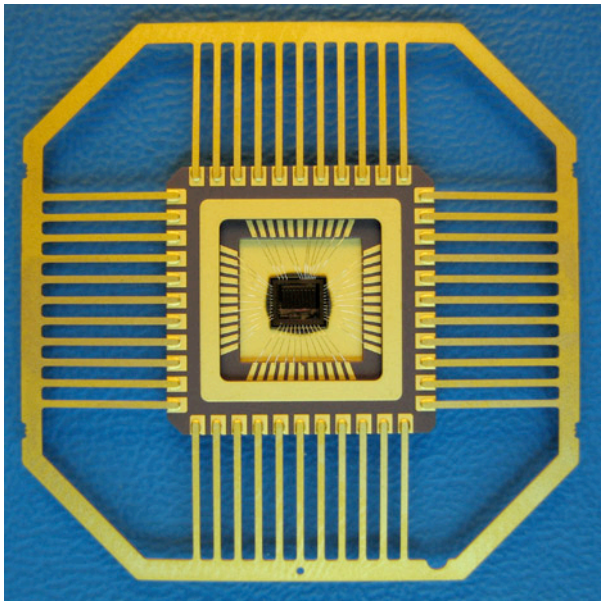

Figure 4: Space Micro's H-Core IC.

The target device (a microprocessor for example) is configured so that a dedicated status signal is periodically asserted to indicate that it is operating normally. Should an SEFI occur in the target device, this periodic status signal would cease to be asserted, indicating a problem. The H-Core chip responds by exercising interrupts to the processor (escalating to higher and higher priority levels) until the microprocessor resumes its normal operation and recovers from the SEFI. (See Figure 5.)

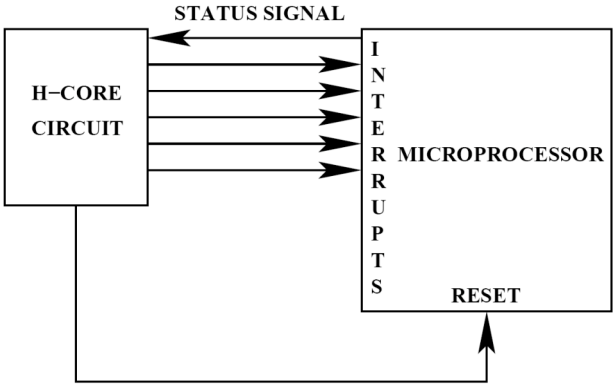

Figure 5. H-Core Connected to Microprocessor

If asserting the microprocessor's interrupt lines does not resume normal operation, then the H-Core can exercise the reset signal and recover the microprocessor in that manner.

The H-Core approach to SEFI mitigation is not just a hardware solution, but also involves software. Software routines are responsible for periodically asserting the status signal to demonstrate normal operation. In addition, the customized interrupt handler routines are written in a way that provides a method of recovery that can bring the processor back to normal operation, or in the extreme, determine that recovery via interrupts is impossible, warranting a full reset.

The effectiveness of H-Core depends in part upon how fast the H-Core detects an SEFI and starts the recovery process. Since there is very little overhead on the part of the processor in asserting the status line (indicating normal, SEFI-free operation), rapid monitoring and detection with minimal effective decrease in microprocessor throughput is easy to achieve.

Once a SEFI occurs, interrupts return the processor to normal operation in less than 100 ns, or alternately return the processor within 5-10 msec when reset is required.

Space Micro successfully demonstrated H-Core operation on several processors when subjected to 51 MeV proton radiation, including Texas Instruments 320C6000 DSP, Equator Technologies BSP-15, Intel's Pentium, and IBM's PowerPC [17].

4. SEE MITIGATION IN FPGAs

In order to mitigate SEEs in FPGAs, the TTMR and H-Core methods are employed. The specific implementation of these must take into account the internal architecture of FPGAs and their particular sensitivity to SEEs.

To detect/correct SEUs, TTMR requires that some type of voting function be performed. An initial assumption would be to implement the voting in the FPGA itself. However, a voter circuit in the FPGA would not be a reliable arbiter of the results since it too would be susceptible to upsets. To solve this problem, the voting function is performed outside of the FPGA by use of a dynamically reconfigurable voter, a VLIW DSP being one approach.

SEFIs in FPGAs result in the FPGA being in an otherwise non-allowed or unstable logic state, and can also result in corruption of the FPGA's control programming. Any SEFI solution for FPGAs must therefore be able to restore the FPGA's programming in such an event. In order to do so, a non-corruptible copy of the programming code must be available. Therefore, the H-Core approach can be expanded to include such capability and this forms the basis of H-Core2.

Hardened-Core2 Mitigation of SEFIs in FPGAs

H-Core2 provides a rad-hard external chip to oversee the FPGA's operation. In the event of an SEFI, H-Core2 is able to perform a readback of the control code to determine if the FPGA's programming has been corrupted. If this is the case, H-Core2 can initiate a "scrubbing" of the FPGA to return it to a known state.

H-Core2 provides the following functions in order to fully support SEFI protection in FPGAs:

- Polling circuitry to periodically check the normal status signals which are sent from the FPGA to indicate normal operation (or, to note the lack thereof which would indicate an SEFI).
- Readback circuitry to read FPGA configuration data to determine if the programming has been corrupted.
- Rad-hard memory to store the FPGA programming code to allow full restoration of the circuit.
- Scrubbing circuitry to reprogram the FPGA as needed.
- FPGA purging control engine to control and coordinate the scrubbing, readback, and any reconfiguration that may be needed.

The response of H-Core2 to fault conditions on the part of the FPGA is in part dependent upon the degree of error the FPGA demonstrates. Simple one-time SEFI errors may be corrected by a partial reconfiguration of the FPGA. A major SEFI (for example, resulting in a unallowable state or full hang in the FPGA) may require a full scrubbing of the FPGA programming.

The degree of an SEFI therefore would need to be interpreted, and a decision of some type made as to whether the FPGA would need a full or partial reprogramming. H-Core2 provides these functions, and additional management is provided by the same processor that is providing the TTMR voting function.

TTMR for SEU FPGA Protection

TTMR is Space Micro's general algorithmic approach towards the mitigation of SEUs. There are variants of this approach however, that allows the system designer to trade-off various factors. These variants can be better understood by detailing the specific parts of a TTMR implementation. Figure 6 shows how two separate modules of logic (identical in function, but using physically separated sets of logic gates) are used to execute a set of operations upon the same input data. The results of each are then fed to an external voter, shown as a Ti DSP processor.

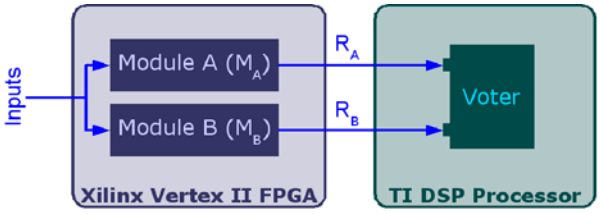

Figure 6. Xilinx FPGA with Ti DSP as Voter

The execution of the logic (upon the inputs in modules Ma and Mb) can be staggered to perform at different times as shown Figure 7. In addition, if the results of Ma and Mb (Ra and Rb) differ, the Voter will indicate such and dictate that these operations are performed again.

Multiple blocks of redundant logic circuitry (execution modules), time staggering, and voting, are the building blocks of TTMR. However, these can be utilized in different ways, resulting in different TTMR algorithms as shown in Figure 8.

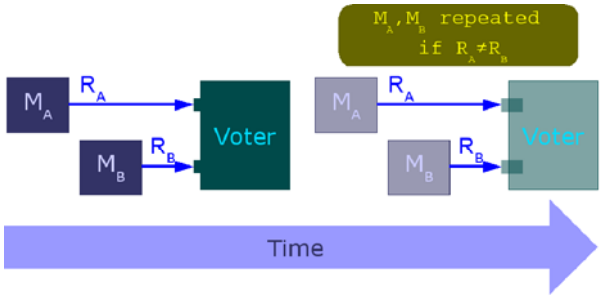

Figure 7. Time Staggered and Repeated Module Execution

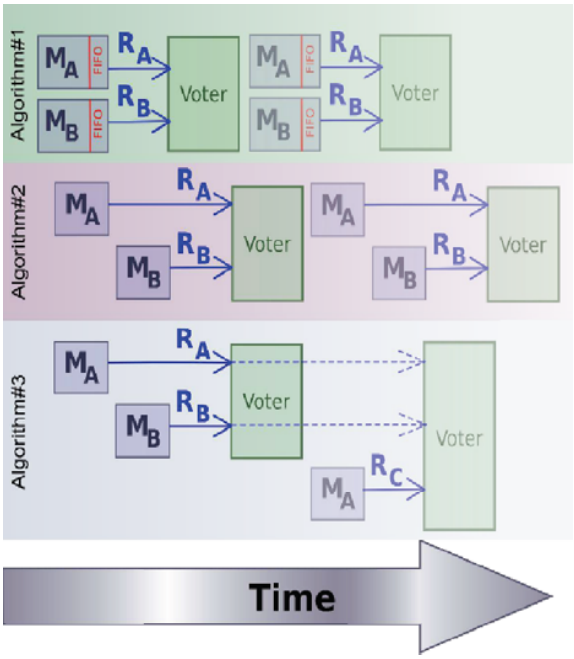

Figure 8. TTMR Algorithms

Algorithm 1 executes a block of logic operations (modules) in succession, and stores the interim results in memory (FIFO). These interim results are then analyzed (for example by creating a checksum) as a block, to determine if any SEU has occurred. If so, then the entire set of logic operations is performed again in order to have sufficient data to create a vote that provides a corrected result.

This algorithm seems to allow for more of the actual target logic operations to be performed per unit time. If the operational environment is one that creates a relatively low number of SEUs, this approach may provide the greatest throughput. However, this is at the expense of using more circuitry and higher gate count, since accommodation must be made to store the interim results and perform the checksum on the data.

Algorithm 2 executes the logic operations (modules) staggered over time. Since these logic operations are checked at each stage (there is not a cue of results stored and compared as a block as in Algorithm 1) this may result in overall slower operation of the target logic. However, the gate count and circuit overhead of such an approach is lessened.

This algorithm seems targeted towards environments with higher expected SEU rates, and/or where gate count should be kept as low as possible.

Algorithm 3 assumes that three full sets of logic operations will be performed each time. The first-stage "voting" of Ma and Mb is not even required. A vote is performed with three full sets of results, and as long as at least two of the results match, that result is used. The third set of operations can be performed on one of the originals modules (Ma in the figure) or a different logic block can be used (Mc, not shown).

This approach introduces further trade-offs. If the operations are always to be performed three separate times, this means either the voter must make provisions for storing the first set of results of the Ma/Mb operations, or else the FPGA circuit design must include an Mc for full parallel execution, which would require a higher gate count.

The algorithms shown are just basic sets of options available in a TTMR-based application. Additional variations are possible. Each of these has its own set of trade-offs that must be considered by the system designer. Each of these may also have implications in terms of its effectiveness in detecting and correcting SEUs.

Combining TTMR and H-Core2

Solutions for solving SEEs in the FPGA requires three elements working in concert, as shown in Space Micro's Fault Tolerant Architecture in Figure 9. The main components of the architecture are:

- Reconfigurable SRAM-based FPGA executing the target design, issuing periodic status signals to the H-Core2 (and sending and receiving data that is part of the target design's normal operation).
- H-Core2 to monitor the FPGA's status signals, detect any SEFIs, store the FPGA programming code, and reprogram the FPGA when (and to the degree which is) required.
- Texas Instruments DSP that performs the voting for TTMR protection of the FPGA, and controls the H-Core2's FPGA programming functions.

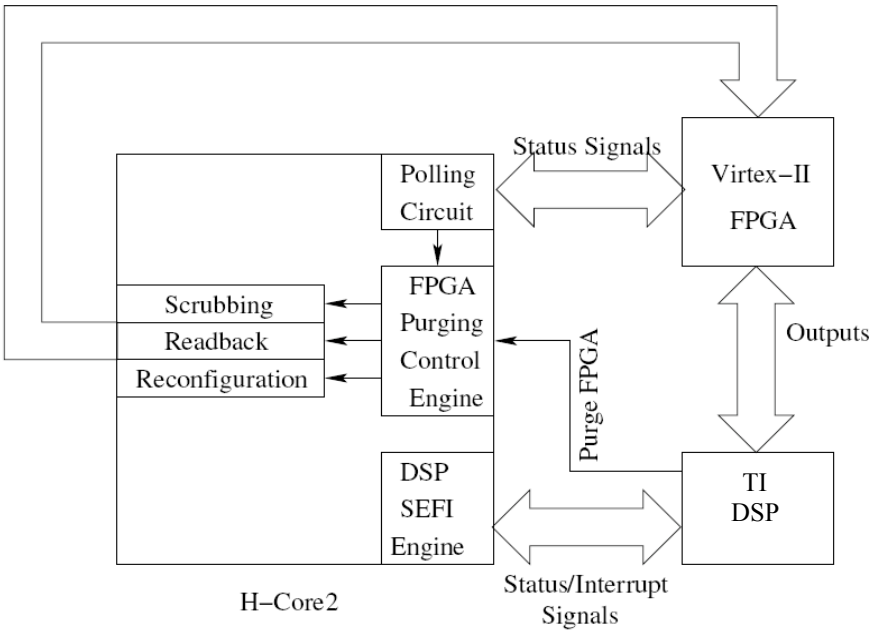

Figure 9. Fault-Tolerant (SEE Correcting) Architecture for Reconfigurable FPGAs

These elements may be combined in various ways. One example of this (see Figure 10) forms the architecture of Space Micro's Proton 300k™ single board computer, which incorporates a TI DSP, Xilinx VirtexII reprogrammable FPGAs, along with H-Core2, TTMR protection, data storage, and various interface options (some of which are not shown).

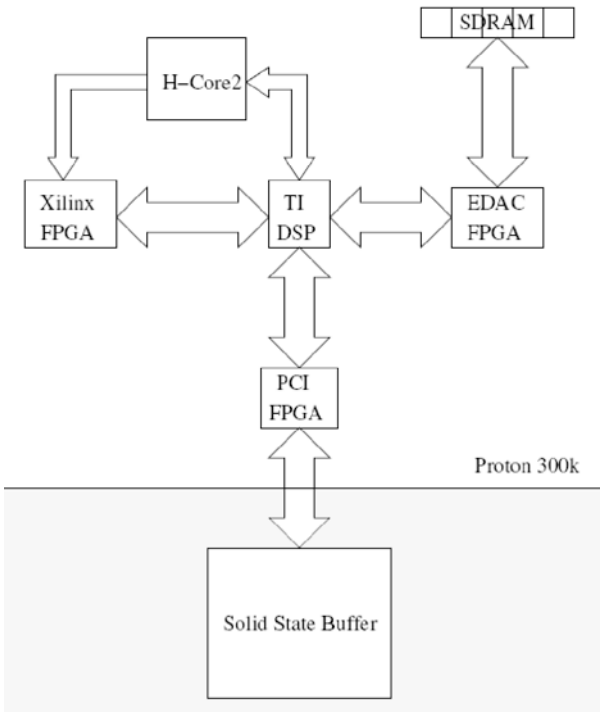

Figure 10. Proton 300k Computer

5. RADIATION TEST RESULTS

A series of tests have been performed to determine the viability of these approaches to mitigate SEEs in reconfigurable FPGAs.

A test board using a 250K gate Virtex-II FPGA and a TI TMS320C6713 DSP microprocessor was used for demonstrating this approach. The TI DSP is a high speed, low power microprocessor, which has shown to have a TID rating of between 100-150 krad.

A proton radiation test on this prototype test bed was performed at Crocker Nuclear Laboratory, University of California, Davis. The test facility hosts a 76-inch Isochronous Cyclotron providing Proton radiation beam, simulating the near space environment by irradiating the FPGA with varying flux performed the test. A beam of 63MeV energy was used for the tests.

A photo of the test setup (at UC Davis) shown in Figure 11 consisted of the test board connected to a control laptop and a SEFI switch in the radiation room. The configuration is shown in Figure 12. The control laptop was remotely accessed using a user laptop outside the radiation room. Since this test is to demonstrate the working of TTMR and H-Core2 in a Xilinx FPGA, only the FPGA was irradiated while shielding the DSP. (Previous successful SEU radiation testing of TTMR and H-Core in VLIW DSPs has been completed [2].)

During the irradiation, a TTMR'd CORDIC DSP algorithm was mapped on the FPGA. Inside of the DSP,

a TTMR'd software voter was operated in order to vote the result.

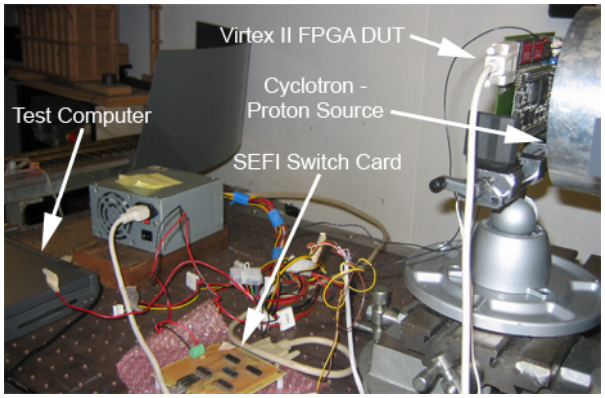

Figure 11. UC Davis Proton Test Set

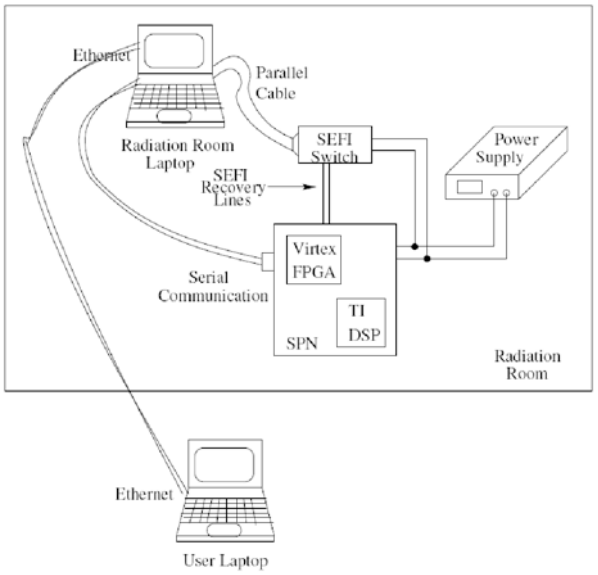

Figure 12. FPGA Radiation Configuration

During the radiation testing, 32 different sequences were completed, showing detection of SEUs in the Virtex II FPGA logic in all test sequences. SEU detection rate was 100%.

The test parameters were such that the number of SEUs in a given period was many orders of magnitude beyond any which would be seen in a real-world application. This high rate overflowed the SEU counter. The SEU counter is necessary in order for the SEU correction mechanism to work properly. So, in some of the test instances, SEU correction was found to not work as expected.

Three variations on the TTMR algorithm were tested.

TTMR Algorithm 1 (dual redundant TTMR with FIFO) was tested in 7 separate sequences, all with successful SEU detection. 2 of the sequences resulted in an overflow of the SEU counter, which is due to the limitations of the test configuration.

TTMR Algorithm 2 (dual redundant TTMR with no FIFO) was tested in 13 separate sequences, all with successful SEU detection. All of the sequences overflowed the SEU counter.

TTMR Algorithm 3 (triple redundant TTMR) was tested in 12 separate sequences, all with successful SEU detection. SEU correction did work properly in these instances.

Some SEUs exhibited behaviors that resulted in SEFI-like conditions. H-Core2 was able to successfully mitigate these.

6. CONCLUSIONS

Time-Triple Modular Redundancy and the H-Core technologies were previously proven to detect and mitigate SEUs/SEFIs in microprocessors. This enabled several advanced, high performance processors to be used in radiation-hardened applications, resulting in the Proton 100k and Proton 200k computers. These same methods have been extended to mitigate SEUs/SEFIs in reconfigurable FPGAs.

The proposed solution for reconfigurable FPGAs offers several advantages over other systems using traditional methods such as TMR. TTMR and H-Core allow the FPGA logic circuit to be run at near full speed, without any extra circuit delays, because results can be summarized into "grouped" answers, where millions of clock cycles can be run and a single answer can be provided to the DSP. This also eliminates the high I/O count required of traditional TMR techniques for Virtex FPGAs. Secondly, use of a DSP as a voter has the advantage of this voter being reconfigurable, since it can be reprogrammed in DSP software. This compares favorably to using a fixed, radiation-hardened voter ASIC, which removes reconfigurability options.

The SEU and SEFI mitigation resulting from TTMR and H-Core for reconfigurable FPGAs provides a very efficient option for satellite applications. These technologies are available in the Proton 300k hardware platform that combines a Ti DSP (rated at 4,000 MIPS with SEU mitigation) along with Xilinx Virtex FPGAs with between 6 and 24 million reconfigurable gates available on board.

7. ACKNOWLEDGEMENTS

This research was supported by NASA Goddard Space Flight Center.

8. REFERENCES

1. Xilinx Single Event Effects, 1'st Consortium Report, Virtex – II Static SEU Characterization.
2. D. R. Czajkowski, M. P. Pagey, P. K. Samudrala, M. Goksel, and M. J. Viehman, "Low Power, High-Speed Radiation Hardened Computer & Flight Experiment," 2005 IEEE Aerospace Conference.
3. J. V. Neumann., "Probabilistic logics and synthesis of reliable organisms from unreliable components," in Automata Studies, Princeton, NJ, Princeton University Press, 1958.
4. D. R. Czajkowski et al., "Ultra Low-Power Space Computer Leveraging Embedded SEU Mitigation," IEEE Aerospace Conference Proceedings, vol. 5, pp. 2315--2328, March 8-15, 2003.
5. P. K. Samudrala, J. Ramos, and S. Katkoori, "Selective Triple Modular Redundancy Based Single Event Upset Mitigation in FPGAs," IEEE Transactions on Nuclear Science.
6. R. Koga et al., "Single Event Functional Interrupt (SEFI) Sensitivity in Microcircuits," Fourth European Conference on Radiation and Its Effects on Components and Systems (RADECS), pp. 311--318, September 1997.
7. J. Howard and K. LaBel et al., "Total Dose and Single Event Effects Testing of the Intel Pentium III(P3) and AMD K7 Microprocessors," IEEE Radiation Effects Data Workshop, pp. 38--47, July, 2001.
8. N. Oh, "Software Implemented Hardware Fault Tolerance," Ph. D Dissertation, Center for Reliable Computing, Stanford University, December 2000.
9. M. P. Pagey, D. R. Czajkowski, P. K. Samudrala, and D. J. Strobel, "SEFI Mitigation Technique for COTS Microprocessors: Demonstration Using Proton Irradiation Experiments," 2004 Military Aerospace Programmable Logic Devices Conference (MAPLD), September 2004.
10. D. R. Czajkowski, "SEFI Mitigation Technique for Microprocessors," 2003 Military Aerospace Programmable Logic Devices Conference (MAPLD), September 2003.
11. P. Shirvani, "Fault-Tolerant Computing for Radiation Environments," Ph.D. Dissertation, Center for Reliable Computing, Stanford University, June 2001.
12. G. M. Swift et al., "Single-event upset in the PowerPC-750 microprocessor," IEEE Transactions on Nuclear Science, vol. 48, issue. 6, pp. 1822--1827, December, 2001.
13. C. C. Yui, G. M. Swift, C. Carmichael, R. Koga, and J. S. George, "SEU mitigation Testing of Xilinx Virtex-II FPGAs," 2003 IEEE Radiation Effects Data Workshop, July 2003.
14. C. Carmichael, "Triple Modular Redundancy Techniques for Xilinx Virtex FPGAs," Xilinx application note: XAPP 197, www.xilinx.com.
15. "Xilinx Single Event Effects," 1st Consortium Report, Virtex-II Static SEU Characterization, www.xilinx.com
16. D. Czajkowski, M. Goksel, P. Samudrala, M. Viehman, M. Pagey, "Radiation Hardened, Ultra Low Power, High Performance Space Computer Leveraging COTS Microelectronics With SEE Mitigation, Military Aerospace Programmable Logic Devices Conference (MAPLD) 2004.
17. D. Czajkowski, P. Samudrala, M. Pagey, D. Strobel, "Low Power High Speed Radiation Tolerant Computer", AIAA/USU Conference on Small Satellites, August 2005.