SEU Mitigation Techniques for Virtex FPGAs in Space Applications

Carl Carmichael[3]
Earl Fuller[2], Phil Blain[1], Michael Caffrey[1]
[1]Los Alamos National laboratory
[2]Novus Technologies, Inc.
[3]Xilinx, Inc.

Abstract

SRAM based logic devices such as FPGAs have some susceptibility to SEU and functional interruption. This paper describes several reliable mitigation techniques for the Virtex series FPGA architecture, which will retain functional integrity while static upsets are detected and corrected.

Additionally, this paper demonstrates how an SEU in an FPGA can be corrected in 3us without disrupting operation of the device, how to build hardened voting circuits, and that a single event has only 1 chance out of 3.25 million of causing a functional interrupt.

I. INTRODUCTION

Re-configurable computing and adaptive hardware is an emerging technology for space applications. The basis for this technology is the capability for device and system level functional changes to be implemented in-system and transmitted remotely. The underlying technology of this capability is Programmable Logic Devices such as Field Programmable Gate Arrays (FPGA).

FPGAs provide an array of logic resources, which may be interconnected, and configured for specific functions. All logic definitions and block connections are controlled by static RAM cells. Thus, this technology is sometimes referred to as "SRAM Logic." The volatility of the functional definition of a configured FPGA allows for on-the-fly reconfiguration of the circuits' functional definition.

The Xilinx XQVR product line is a radiation-tolerant version of the of the commercially popular Virtex series FPGA. Virtex has become a common ASIC replacement in commercial markets due to its density, performance, and wide range of capabilities. The XQVR utilizes an epitaxial process that renders it latch-up immune to an LET of 125MeV-cm^2/mg. Although the XQVR is latch-up immune, the SRAM cells do have some susceptibility to Single Event Upsets (SEU). This paper intendeds to provide methodologies for detecting, correcting, and mitigating such SEUs.

II. ASICS VS. VIRTEX FPGAS (XQVR)

ASIC devices provide the advantages of "instant-on" and some level of SEU tolerance depending on the process technology employed. They do not require an initial configuration cycle before becoming functionally active after power-up, and their logic configuration is not altered by SEUs. However, ASICs have a disadvantage in that their functionality may never be altered. As a result, not only may it never be updated, upgraded or corrected in any way, but any permanent destruction of its sub-circuits, such as Singe Event Gate Rupture (SEGR), renders that portion of the circuit forever disabled.

An FPGA requires a configuration cycle after power-up to be functionally active. However, the Virtex FPGA utilizes a high-speed configuration interface (discussed later in greater detail) which can bring the device functionally active within 20ms (or less) after power-up. The configuration interface also provides a fast and efficient method for the detection and correction of static upsets to the configuration memory. Additionally, the Virtex architecture provides logic elements to implement SEU mitigation circuits whose functionality are not dependent on SRAM cells. Therefore, there are simple design techniques which may be utilized to render the device nearly impervious to SEU effects.

The re-programmability and readback capability of the Virtex FPGA not only provides for fast detection and correction of SEUs, but also allows for the functionality of the device to be altered an unlimited number of times which allows for functional evolution throughout the mission life span. Additionally, the mission life span itself may be increased by this capability. Although no SEGR effects have ever been observed during testing, if any portion of the logic array is ever rendered inoperable, the required functional design can easily be re-implemented to not utilize the effected area of the device and its new bit-stream can be remotely transmitted to the spacecraft/application.

Another important advantage of re-programmable logic is that [they] are essentially commercial-off-the-shelf products (COTS). Any long-term usage of an ASIC product is susceptible to eventual system redesign due to diminished material supply (DMS).

III. High Speed Virtex Configuration

The Virtex series FPGA provides multiple access ports for the purpose of writing and reading data to/from the configuration memory array. One such access port is the SelectMAP (Selectable Microprocessor Access Port) interface. SelectMAP is an 8-bit parallel bi-directional synchronous interface to the configuration control logic. All aspects of the configuration control logic and configuration memory can be addressed and manipulated through the SelectMAP interface.

The functionality of a specific user design is represented by a bitmap of binary data called a “bit-stream”. The process of configuration is to load the bit-stream into the FPGA (sometimes referred to as “download”). This will bring the FPGA functionally active with the user’s defined functionality. The size of a bit-stream is constant for a given part (1.7 Mbit for a V300, 6.27 Mbit for a V1000). The SelectMAP interface has a maximum speed of 66 Mbyte/s (528 Mbit/s). At clock speeds of 50MHz and below, no handshaking of the data is required. Therefore, all further calculations will be made assuming a thru-put of 400 Mbit/s. Since a V1000 (1 million system gates) has a ~6.5 Mbits configuration memory array, this device may be configured in ~20ms without interruption.

The SelectMAP interface may also be used to read the configuration data back out of the device at the same rate. This capability (referred to as “readback”) allows the current configuration contents to be verified against expected data. Thus, this is a useful detection scheme for configuration memory cells that have been altered by SEUs.

The Virtex configuration interface has the additional capability of addressing small portions of the configuration memory map for read and write operations. This is referred to as “partial configuration”, and provides an extremely efficient means for SEU correction.

A quick overview of the Virtex architecture is needed before discussing SEU detection, correction, and mitigation. For further explanation of the SelectMAP interface and configuration operations refer to Xilinx Application Note XAPP138 “Virtex FPGA Series Configuration and Readback”.

IV. Architecture Overview

Virtex devices feature a flexible, regular architecture that comprises an array of configurable logic blocks (CLBs) surrounded by programmable input/output blocks (IOBs), Figure 1, all interconnected by a hierarchy of fast, versatile routing resources. The abundance of routing resources permits the Virtex family to accommodate even the largest and most complex designs.

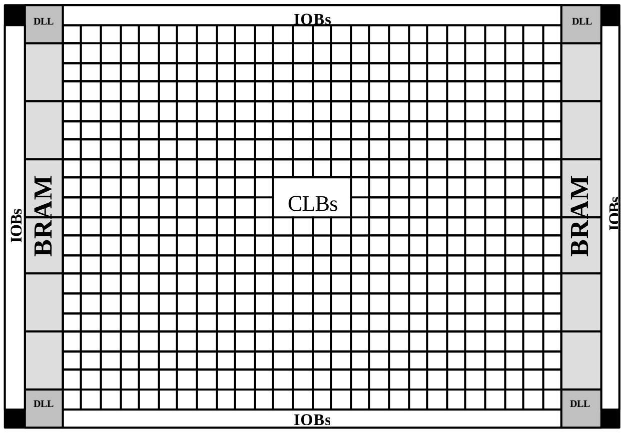


Figure 1: Virtex Array Architecture

A. Virtex Array

The Virtex user-programmable gate array comprises two major configurable elements: configurable logic blocks (CLBs) and input/output blocks (IOBs).

- CLBs provide the functional elements for constructing logic
- IOBs provide the interface between the package pins and the CLBs

CLBs interconnect through a general routing matrix (GRM). The GRM comprises an array of routing switches located at the intersections of horizontal and vertical routing channels. Each CLB nests into a VersaBlock™ that also provides local routing resources to connect the CLB to the GRM.

The Virtex architecture also includes the following circuits that connect to the GRM.

- Dedicated block memories (BRAM) of 4096 bits each.
- Clock DLLs for clock-distribution delay compensation and clock domain control.
- 3-State buffers (BUFTs) associated with each CLB that drive dedicated segmented horizontal routing resources.

The basic building block of the Virtex CLB is the logic cell (LC). An LC includes a 4-input function generator, carry logic, and a storage element. The output from the function generator in each LC drives both the CLB output and the D input of the flip-flop. Each Virtex CLB contains four LCs, organized in two similar slices, Figure 2.

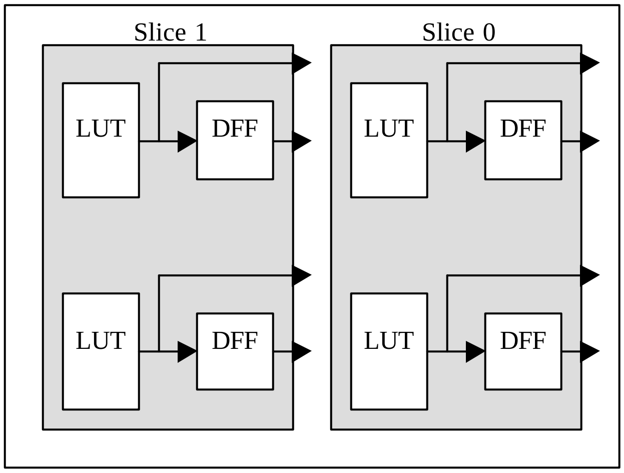


Figure 2: Virtex Configurable Logic Block

In addition to the four basic LCs, the Virtex CLB contains logic that combines function generators to provide functions of five or six inputs. Consequently, when estimating the number of system gates provided by a given device, each CLB counts as 4.5 LCs.

Virtex function generators are implemented as 4-input look-up tables (LUTs). In addition to operating as a function generator, each LUT can provide a 16 x 1-bit synchronous RAM. Furthermore, the two LUTs within a slice can be combined to create a 16 x 2-bit or 32 x 1-bit synchronous RAM, or a 16x1-bit dual-port synchronous RAM.

The Virtex LUT can also provide a 16-bit shift register that is ideal for capturing high-speed or burst-mode data. This mode can also be used to store data in applications such as Digital Signal Processing.

Each Virtex CLB contains two 3-state drivers (BUFTs) that can drive on-chip busses. Each Virtex BUFT has an independent 3-state control pin and an independent input pin both with a selectable inversion.

The output connections of the buffers, shown in Figure 3, select from four horizontal bus channels. In each CLB column one of the four channels may be selected to either terminate or continue to another segment. This bus architecture allows for multiple bus structures of varied sizes to be implemented in the same CLB row.

For a complete description of the Virtex logic elements and features please refer to the Virtex Data Sheet.

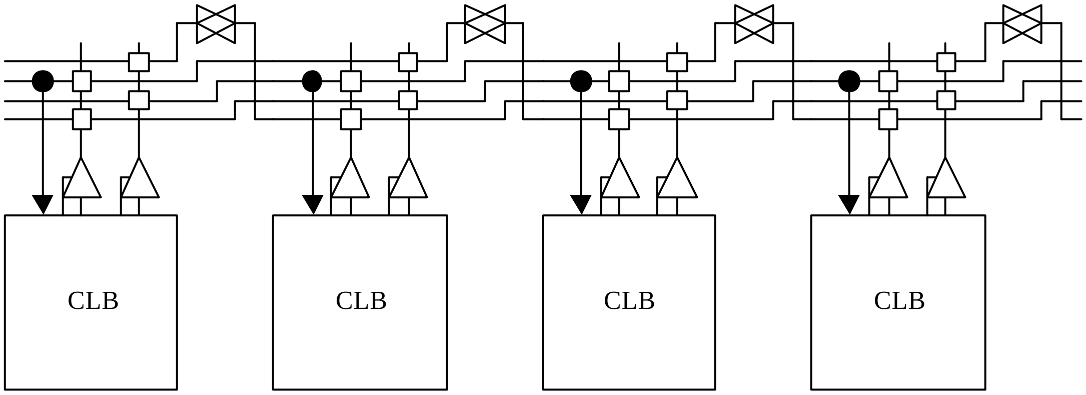


Figure 3: Horizontal 3-state Bus Structure

B. Configuration Memory

Values stored in static memory cells control the configurable logic elements and interconnect resources. These values load into the memory cells on power-up, and can reload if necessary to change the function of the device.

The configuration memory cells lie closely to the specific functions they control and are laid out in a regular pattern. A data-frame is a 1-bit slice of the memory array along the vertical axis. The configuration data is written to the configuration memory from configuration registers one data-frame at a time. Therefore, the smallest portion of configuration data that may be read from, or written to, the configuration memory is one data-frame.

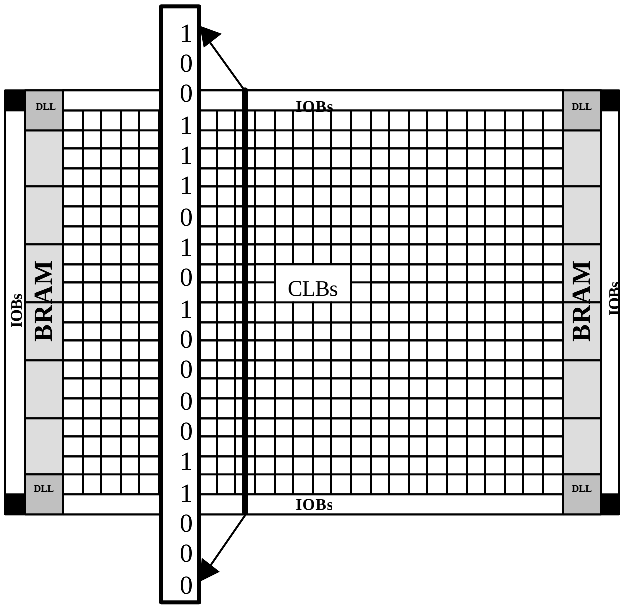


Figure 4: Configuration Memory Data Frame

Shown in Figure 4, a single data-frame contains portions of configuration data for each and every block that lies in that column. Hence, multiple data-frames are required to describe the complete width of a column.

In order to read and write individual data-frames, each must be uniquely addressed by the configuration logic. Therefore, each column is identified by a "major address" and each frame in that column is identified by a "minor address."

In the context of data-frames, there are five column types:

- 1 Center Column (8 frames)
- *n* CLB Columns (48 frames)
- 2 Block RAM Interconnect Columns (27 frames)
- 2 Block RAM Columns (64 frames)
- 2 IOB Columns (54 frames)

The number of CLB frames depends on the device size as does the bit size of the data-frame itself. The center column includes all the global clock structures and elements. The CLB columns include the IOBs along the top and bottom of the die. Similarly, the Block RAM Interconnect columns contain the DLL information.

For a more detailed examination of the Virtex configuration logic architecture, please refer to Xilinx Application Note XAPP151 "Virtex Configuration Architecture Advanced Users' Guide."

V. SEU DETECTION AND CORRECTION

As mentioned before, the readback function is an efficient means for SEU detection. If a particle penetrates the susceptible portion of a configuration SRAM cell and thus alters its state, a readback and verification of the configuration data will detect the upset. To perform a verification (SEU detection), the configuration data is readback from the device and compared to the configuration bit-stream. Not all of the data readback from the device is applicable to direct comparison. Therefore, some configuration data bits must be masked from this process. Most commercial applications utilize a mask file for this function (mask files are produced by Xilinx S/W upon user request via a bit-stream generation option). A mask file provides a bit-for-bit indication, for the entire bit-stream, whether that bit should be compared or ignored. However, for space applications where memory is expensive and board space is premium, storage of an extra 6.5 million bits is greatly undesirable. Therefore, a more efficient means is required.

For a currently developing application at Los Alamos National Laboratories, involving a low earth orbit satellite, an alternative methodology for verification has been developed. The mask file has been reduced to an algorithm embedded in the configuration and readback controller, and the actual bit-stream may be further reduced with a compression algorithm. A dedicated device performs a continuous verification of the Virtex FPGAs' configuration memory for upsets. Whenever an upset is detected, it is immediately corrected through a configuration operation. The projected low earth orbital path for this application is expected to see upsets at a rate of 1 per hour spread across three devices. Since detection and correction is always completed within 40ms for each device, there should typically be 90,000 detection/correction cycles (or 180,000 detection cycles assuming no other upsets) between statistically expected upsets. This means that the device is reliably operating without upsets or interrupts 99.9989% of the time.

The time required for SEU correction may be dramatically decreased by the use of partial configuration. The above calculations assumed a correction time of 20ms. This implies that the correction of SEUs involves complete re-configuration of the device. This is a significant point of consideration because complete re-configuration implies "de-configuration" which means bringing the part "off-line" during the correction cycle and thus losing all internally stored data. Not only is this undesirable, but is in fact completely unnecessary. Partial configuration allows individual frames to be written to the configuration memory. Therefore, only the frame that contains the SEU effected cell would need to be corrected. Assuming that only a single data frame needed to be loaded, the correction time now falls to a mere 3µs.

Aside from the efficiency and speed of SEU correction with partial configuration, a far more important consideration is the fact that the device may be left completely active during the correction cycle. Beside the obvious desire to not lose the current logical state and internally stored data, this is statistically unnecessary. That is, in the event of a single upset there is a very high probability that the effected cell does not even effect the basic functionality of the configured design. This became evident upon calculation of the effective cross-section of SEU susceptibility with respect to the density of storage elements in the device.

The V1000 has approximately 6.27 million storage elements that are susceptible to upsets with an average (weighted) cross section of 8e^{-8}cm^{2}. This represents approximately ½ of the total die area. Approximately 85% of these elements control routing pips. A typical design (80~90% utilization) uses less than 10% of the routing pips. Therefore, a particle that hits within the ~1cm^{2} die area has at least a 76.5% chance of hitting a routing pip memory cell that it will not cause a functional interrupt.

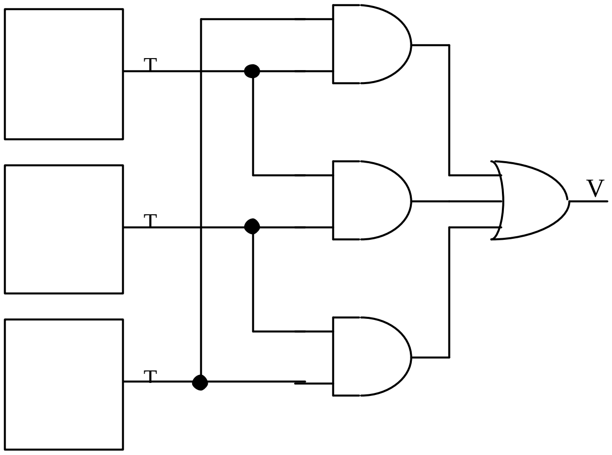
An additional method of SEU detection could be for the FPGA to signal the host system when an upset occurs. This can be done without the use of readback and provides the additional capability of identifying SEFI, or transient upsets, which readback and verification would be oblivious to.

A simple detection scheme is to duplicate internal logic and compare similar outputs. When similar outputs differ from each other, then an upset has occurred. This may be used to signal a device, that either selects the outputs from redundant devices or initiates detection/correction cycles, that an upset has been detected. This scheme has certain advantages over other types of mitigation schemes and is discussed further in the SEU Mitigation section of this paper.

VI. SEU MITIGATION

In some systems SEU detection and correction alone can achieve an acceptable level of reliability. However, for applications where an even higher level of reliability is needed, or simply that any interrupt in service is unacceptable, SEU mitigation techniques may be applied. A good SEU mitigation technique should filter out the effects of upsets, during their short existence, as well as filter out the results of transient upsets or other SEFI effects.

A commonly known method for SEU mitigation is "triple module redundancy with voting." This mitigation scheme uses three identical logic circuits performing the same task in tandem with corresponding outputs compared through a majority vote circuit. A simple example of this is shown in Figure 5.



SEU Mitigation Truth Table

TR0	TR1	TR2	V
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure 5 : Triple Redundancy with Voting

Most SRAM based logic devices cannot reliably implement this function because the voting circuit itself would have to be implemented in SRAM cells just as any other boolean function would be, and is therefore itself equally susceptible to upsets.

The Virtex architecture provides a perfect solution to implementing this circuit reliably. The Tri-State Buffers (BUFTs), described in the architecture overview, are in fact not actually pass transistors. They are actually a hard-wired AND-OR logic structure similar to that shown in Figure 6 (all actual logic gates in this structure are two-input gates).

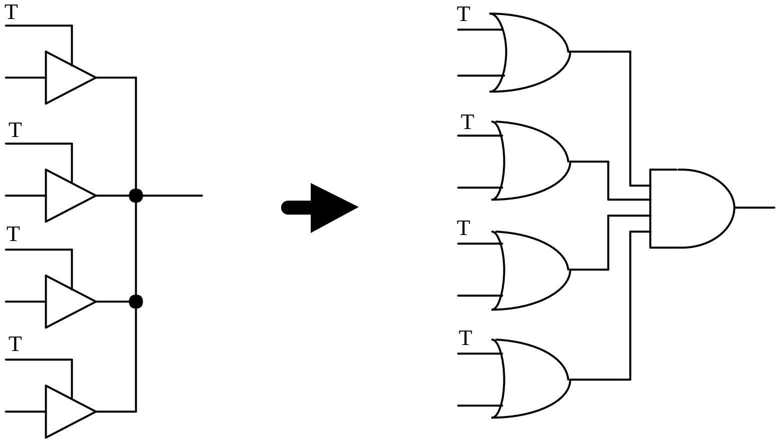
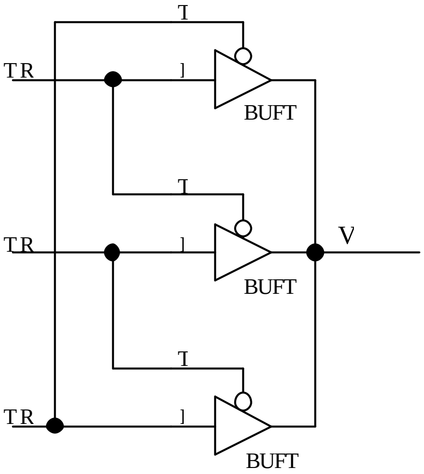


Figure 6: Virtex BUFT Structure

These elements can be cross-connected to produce the same boolean function as that needed for the majority vote circuit. This structure is shown in Figure 7. Using these elements in this fashion provides a voter circuit whose functional description is not based on the contents of any SRAM cells, which may get upset. The only aspects of this circuit which are controlled by configuration memory cells are the routing pips which connect them together. Upsetting one of these cells would only result in temporarily disconnecting one of the inputs or outputs of one of the BUFTs. Such an upset would not effect the output of the voter circuit. In fact, this method is completely impervious to a single upset failure. Only multiple simultaneous upsets would cause this function to fail.



TRV Truth Table

TR0	**TR1**	**TR2**	**V**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure 7: Voting circuit with BUFTs

Not only would multiple upsets be required to cause a failure in this circuit, these upsets would have to occur in very specific patterns. The probability of such upsets becomes of far less concern. Even in a rad-hard ASIC, such a mitigation scheme is only reliable when no more than one out of the three signal nodes to be evaluated are upset. If two or more of the redundant modules are presenting an incorrect result due to multiple upsets, then a properly functioning voter circuit would correctly favor the incorrect data. Since this is far more probable than upsetting this voting circuit scheme to the point of failure, it should be concluded that this circuit exceeds a reasonably expected level of reliability. Therefore, to further increase overall reliability of a functioning system, our attention should be directed to further mitigation of the results either presented to, or obtained from, any particular mitigated node. The following sections provide mitigation implementation examples of varied complexities along with their associated trade-offs, advantages and disadvantages.

A. Module Redundancy and Mitigation

A very simple method for implementing SEU mitigation in a users' FPGA design is to replicate redundant instances of an entire module and mitigate the final outputs of the modules. This is demonstrated in Figure 8.

In this case a module may represent either the entire design for a particular device or a sub-component of that design. This is a very effective means of SEU mitigation that is easy to implement and can be performed entirely within a single device as long as the user's design does not utilize more than 1/3 of the total device. Any design that fits into the CLB array of a V300, can be tripled and mitigated in a V1000. However, care should be taken with the utilization of certain other elements within the Virtex architecture.

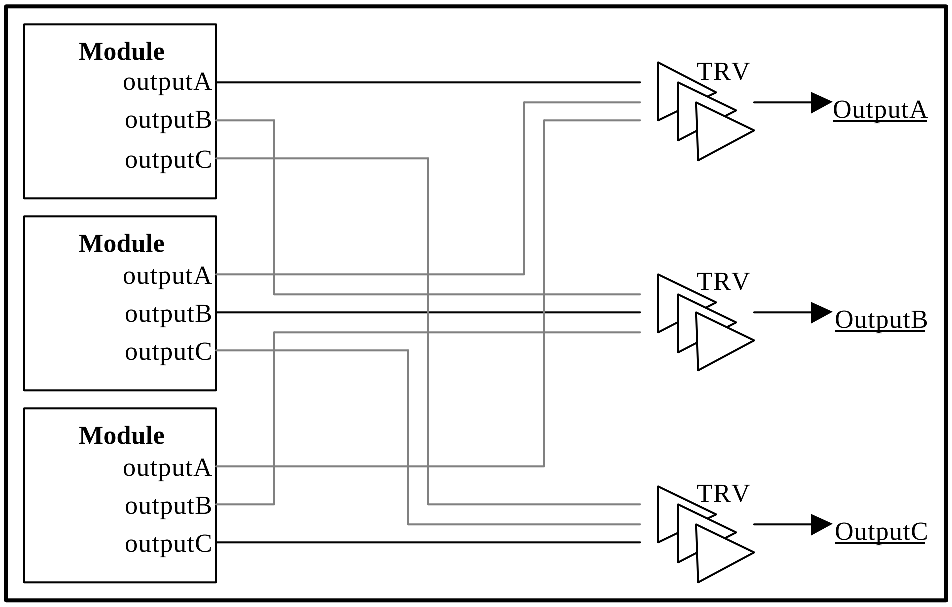


Figure 8: Module redundancy and mitigation

A V1000 has only twice as much BlockRAM as a V300, and all Virtex devices have the same number of DLLs and clock buffers regardless of device size. The I/O utilization, however, does not increase with the use of redundancy. Therefore, the initial design may be targeted for the array size of a V300, but fully utilize the I/O count of a V1000. This could be considered an added return for the cost of redundancy.

In the event that the initial design consumes more than 1/3 of the resources of the largest available device (V1000), then the designer should consider alternatives such as logic partitioning, logic duplication, or device redundancy.

The clear advantages to this example of module redundancy is that it may be contained within a single chip solution (an important cost advantage) and will not impact system performance. The obvious disadvantage is the limitation on the design size (less than 1/3 of the total device).

B. Logic Partitioning for Mitigation

In the case where the total design is more than 1/3 of the device size, the design could be partitioned into modules small enough to be replicated and mitigated within a single device, and spread across several devices. This is demonstrated in Figure 9.

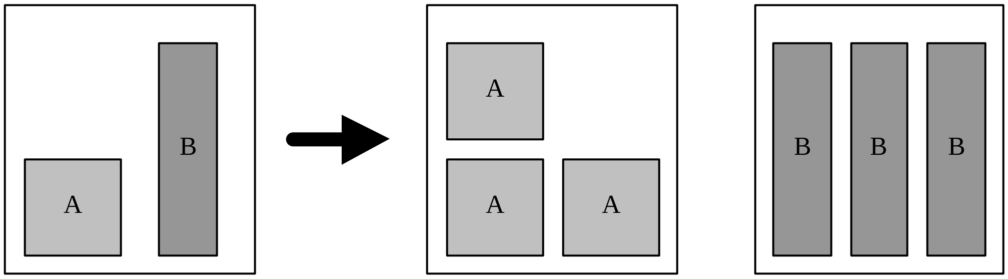


Figure 9: Module partitioning

As shown in Figure 9, module A is replicated and mitigated in one device while module B is replicated and mitigated in another. All outgoing signals, whether they are internal signals between the modules or design outputs, should be mitigated before going off chip. While this may somewhat complicate the performance constraints and specifications, it should not impact the achievement of the needed performance as long as care is taken to not stretch the critical paths of the design across multiple chips.

In the case where the total design is larger than a single device, requiring multiple devices for the implementation, logic partitioning becomes intuitive. The advantage to this method is that no external mitigation is needed between the separate FPGAs as would be needed with device level redundancy. This method does, however, represent an added cost not only for the multiple FPGAs, but for the increased board space utilization as well.

C. Logic Duplication and Mitigation

In the case where the design is less than ½ the size of the total device, an alternative to logic partitioning is logic duplication. This concept was briefly mentioned in the SEU Detection section earlier. If logic is duplicated and like outputs compared, whenever one set of outputs differ an SEU or SEFI has been detected.

In Figure 10, duplicate modules, A and A', are duplicated again in a second device. Each output is disabled whenever an upset is detected for that path allowing the unaffected device to continue driving the data line with the correct value. The cross-connected active High enabled BUFEs in each device implement an XNOR function, which drive the active Low enable of the OBUFT.

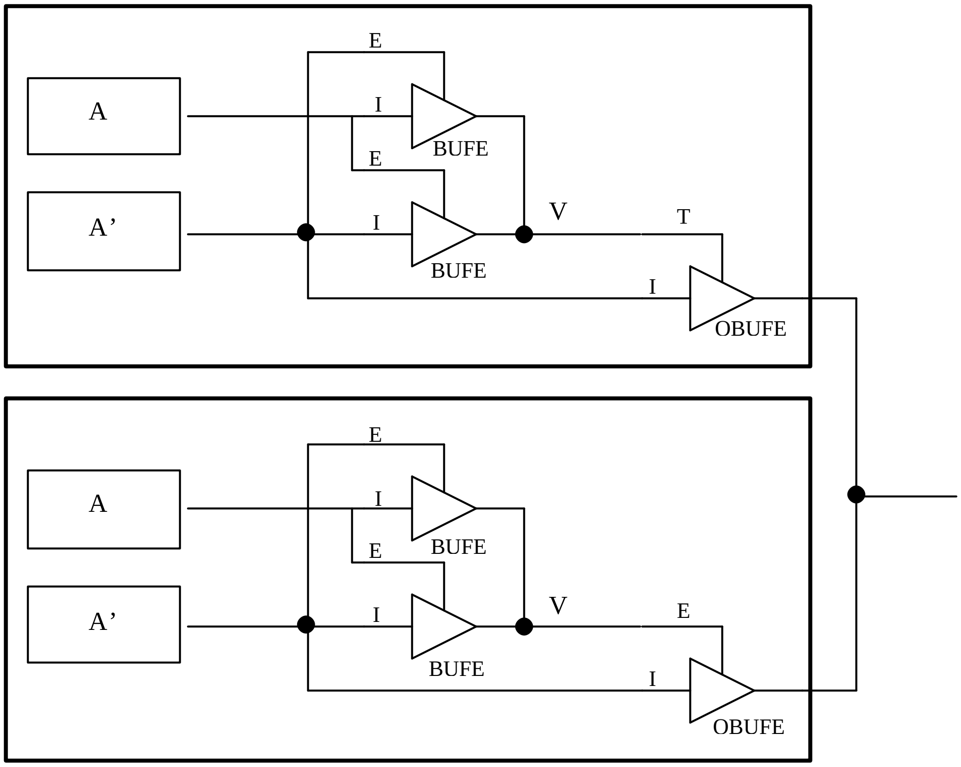


Figure 10: Dual Voting Double Redundancy

One advantage to this method is that it is a form of device redundancy without the need for any external mitigation devices. This is significant because in the case of a total device failure the redundant device would continue processing. Total device failure is an extreme improbability, but not completely impossible. This condition is discussed in the "Single Event Functional Interrupts" section. In the case of a total device failure, all previously discussed single device mitigation methods would be susceptible to a momentary disruption of service. Another advantage is that in the absence of upsets, both output drivers are active, effectively doubling the drive strength on that trace. However, the disadvantage of this may be additional board noise due to skew in the output transitions times.

D. Device Redundancy and Mitigation

Triple device redundancy and mitigation is the most rock-solid mitigation method. This is shown in Figure 11. It has the highest reliability for filtering single and multiple event upsets, multiple transient upsets, and any other functional interrupts including total device failure. However, this is also the most costly solution and provides only a marginal actual improvement over alternative methodologies.

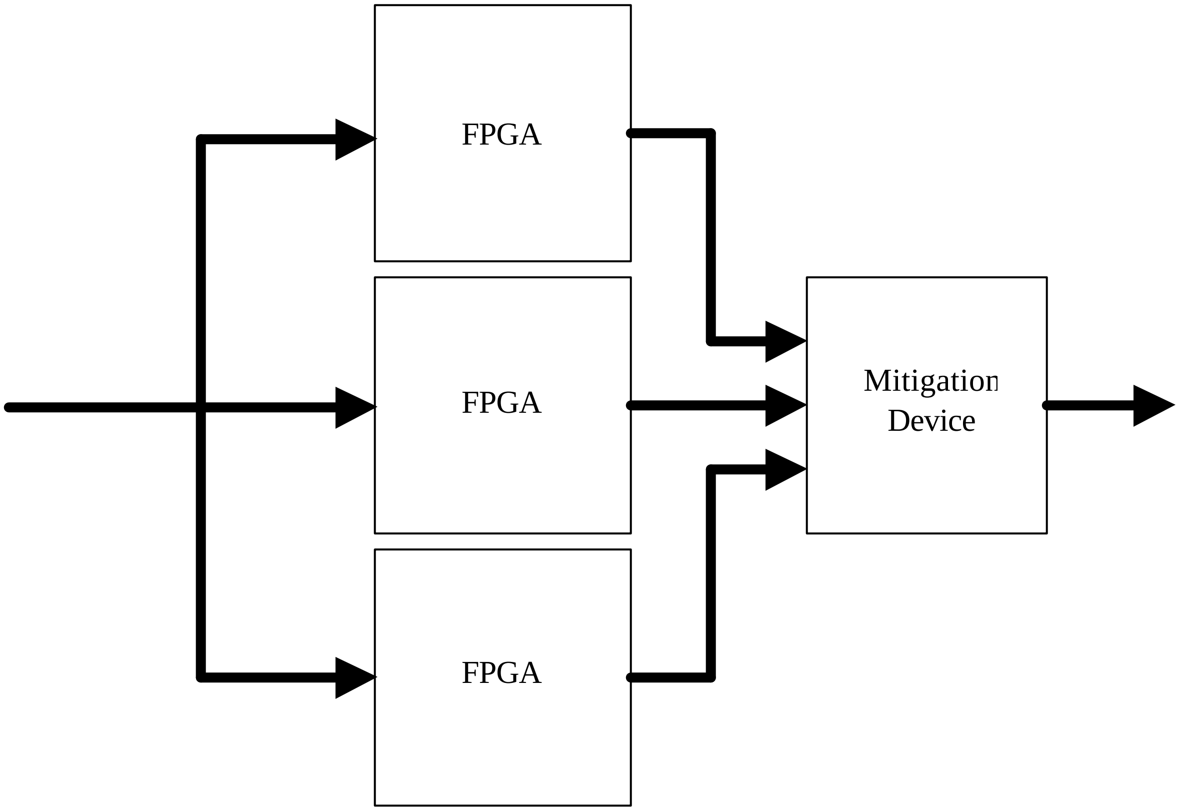


Figure 11: Triple device redundancy

In Figure 11, a single FPGA design is replicated three times in redundant FPGA devices. The mitigation of the redundant devices requires a fourth device (possibly more depending on I/O count). The mitigation device could either be another programmable logic device with internal redundancies, or a small rad-hard ASIC.

Alternatively, a processor could be used to manage the redundant devices as a queue. Whenever an upset is detected in one of the devices, [that] device is taken offline and repaired while another device is selected from the redundancy queue to continue processing.

VII. Single Event Functional Interrupts

There are certain single event effects which may cause a complete functional interruption. This is why only device redundancy will make a system completely impervious to a single event functional interrupt.

An ion has 1 chance out of 13 million to hit a specific storage cell of a V1000, and only a small number of specific cells can cause a complete functional interruption. Additionally, the worst-case result of such an interruption is merely that the device would need to be re-configured.

A. Device De-configuration

The Power-On Reset (POR) circuitry contains three SRAM cells and one flip-flop register that signal when a successful power-up has completed. This signal will initiate an initialization process, which clears configuration memory to prepare the device for configuration.

Upsetting one of these four storage elements will re-initiate the initialization process requiring that the device be re-configured.

This phenomenon was observed during heavy ion testing at the Texas A&M cyclotron facility. However, this condition was only observed at a fluence above 10^5 ions/cm^2.

B. Interruptions from JTAG Operations

The JTAG/Boundary-Scan circuitry has a standard susceptibility similar to that present on any device technology which utilizes this functionality.

The standard TAP controller implementation is a 4-bit binary encoded state-machine. A single event upset to one of these registers can move the controller to any of the available TAP states. This carries the possibility of activating the boundary-scan registers and disengaging the I/Os from standard operation.

The recover mechanism for this condition is to hold the TMS input in it's return state (Logic High) while driving the TCK with a high speed free-running oscillator. This will insure that should the TAP controller jump to an unwanted state it will never be more than 5 clock cycles away from returning to the Test-Logic-Reset state. The maximum rated frequency for the JTAG clock input (TCK) of a Virtex device is 33MHz. Therefore, the worst-case recovery time would be 152ns.

The Only JTAG state that would cause all the I/O pins to become outputs is the EXTEST instruction. If the 5-bit JTAG shift register contained all zeros <00000> and the TAP controller jumped to the UPDATE-IR state, then the EXTEST function would be activated. However, if the instruction register was preloaded with all ones <11111> with a *shift-IR* operation during device initialization immediately after power-up, then no single event could possibly cause any undesirable conditions. In fact, it would take six very specific events for this condition to still occur.

C. Activating Output Drivers on an Input Pin

For any given single input multiple configuration cells must be upset to activate the output driver for a single IOB. Though this condition is extremely unlikely (and perhaps beyond the scope of an SEFI), such a condition could cause some contention, but will not damage the device.

VIII. Conclusions

Triple device redundancy is a proven SEU mitigation technique for non-volatile ASIC products. For SRAM based FPGAs, the addition of rapid detection and correction along with internal mitigation circuits make the Virtex FPGA as SEU immune as any technology on-orbit while at the same time offering performance, features, and capabilities that were never before available.

REFERENCES

[1] Fuller, E., et al, "Radiation Test Results of the Virtex FPGA and ZBT SRAM for Space Based Reconfigurable Computing," MAPLD Paper, C-2.

[2] Xilinx, "The Programmable Logic Data Book," ©1991. http://www.xilinx.com/partinfo/virtex.pdf