

LA-UR-02-6138

*Approved for public release;
distribution is unlimited.*

Title: Single-Event Upsets in SRAM FPGAs

Author(s): Michael Caffrey, 111180, NIS3
Paul Graham, 181856, NIS3
Eric Johnson, 187949, NIS3
Los Alamos National Laboratory
Michael Wirthlin, BYU
Carl Carmichael, Xilinx

Submitted to: Military and Aerospace Applications of Programmable Logic
Devices (MAPLD)
Laurel MD, USA September 2002

Los Alamos

NATIONAL LABORATORY

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Single-Event Upsets in SRAM FPGAs

Michael Caffrey¹, Paul Graham¹, Eric Johnson^{1,2}, Michael Wirthlin², Nathan Rollins², Carl Carmichael³

¹Los Alamos National Laboratory, Los Alamos NM, USA

²Brigham Young University, Provo UT, USA

³Xilinx Inc., San Jose CA, USA

Abstract—Field Programmable Gate Arrays (FPGAs) are indisputably useful for space missions where system schedule and cost are critical but production quantity is low. SRAM-based FPGAs are uniquely suited for remote missions because of the ability to change function *in situ* and because they offer substantial signal processing performance. Single Event Upsets (SEUs) are of utmost concern for SRAM FPGAs because the logic functions themselves are sensitive to unintended change. This paper discusses our work with the Xilinx Virtex FPGA and the current understanding of the device’s sensitive cross-section. Also discussed are considerations for SEU detection, and methods for reducing SEU sensitivity and increasing SEU observability.

Index Terms—FPGA, Reconfigurable Computing, Radiation Tolerant, Remote Sensing, Satellite Instrumentation.

I. INTRODUCTION

We intend to use the Xilinx Virtex SRAM-based FPGA in an orbital study of the ionosphere. The system will detect and characterize events in a high-speed radio frequency (RF) channel. Our reliability requirements may be unique to our system but the description of upsets and our detection and mitigation efforts should be useful in a broad range of

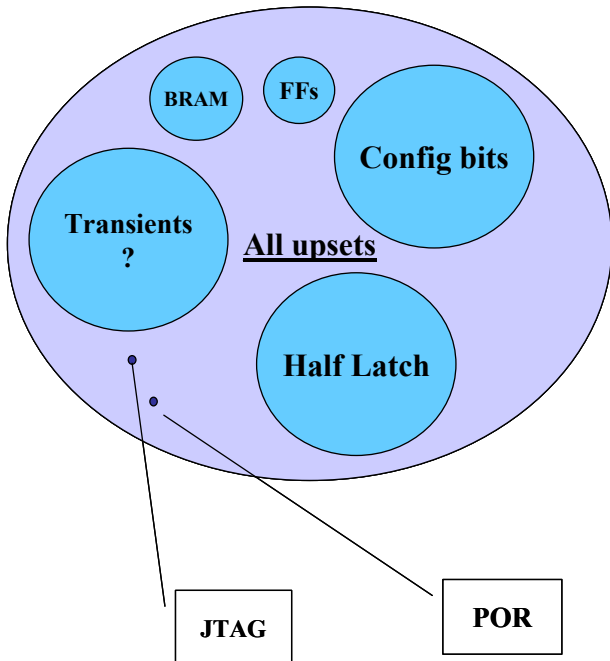


Figure 1: The set of all upsets categorized for observability/mitigation considerations.

applications. For our application we want to achieve the most reliability possible for low system cost (throughput, dollars, logic density, etc.), zero effective upsets is not our objective. Others have considered the path to zero upsets [2][7]. Our goal is to make as many of the FPGA resources available to the processing algorithm as possible and still achieve the reliability required for our system. The consequences of upsets will have variable severity depending on where in the device they occur. This paper describes what we know about device-upset categories and what device- and system-level techniques can be used to increase observability or mitigate risk. Unresolved questions are also mentioned.

II. SENSITIVE CROSS-SECTION

The set of all upsets includes categories with different consequences and different cross-sections. Figure 1 indicates the various categories we suspect exist for the Virtex FPGA. Most categories (flip-flops, JTAG TAP controller, Block SelectRAM, transients) are common with many digital devices. The Virtex has unique sensitivities such as the configuration bitstream, half-latches, and the configuration

Table 1: Relative size of contributors to the Virtex XQVR1000 sensitive cross-section.

User Flip-Flops	26,112	0.4%
LUT Bits	393,216	6.4%
Block SelectRAM	131,072	2.1%
Configuration	5,603,456	91.0%
Single Event Functional Interrupts	?	<.0021%
Transients	?	?
Half-latches	?	?

management controller (which is referred to as POR—Power On Reset—in Figure 1 due to its reset register sensitivity). Heavy-ion testing [1][3] has shown that the average saturation cross-section per bit in Table 1 is $8E-8 \text{ cm}^2$ and that the cross-section measured for the Single Event Functional Interrupts (SEFIs) is $1E-5 \text{ cm}^2$ total per device. Those results should scale from the 300,000-gate device tested previously to the 1-million-gate device discussed here. The relative contribution from half-latches is difficult to measure because they are not directly observable. However, a mitigation scheme to eliminate the half-latch contribution is discussed later in the paper.

The relative percentages called out in Table 1 are in reference to the total static-test cross-section, not including the contribution from half-latches and transients, if they exist, which only manifest themselves in dynamic testing. The

lookup table (LUT) and configuration bits are both observable in the configuration readback bitstream, which the device can provide through the SelectMAP configuration interface while the design is in operation. This makes the vast majority, more than 97%, of static upsets directly observable while the system is in service. For our purposes, LUT bits are implicitly included when discussing configuration bits unless specifically noted.

A. Transient Effects

The existence of transient induced upsets has not been established. Proton accelerator experiments [3] suggested that approximately half of all upsets detected during dynamic testing were not due to configuration bitstream upsets. Those experiments failed to show any detectable clock rate dependence that would suggest the presence of transient effects. There may be an unknown contributing cross-section or the test may have been faulty. The test had drawbacks, for instance, half-latches had not been removed from the designs; in fact, the test helped prompt their discovery. In addition, the design was not a ‘golden chip’ test but, due to the complexity of the test fixture, the design under test was self-checking. An upcoming experiment will revisit the issue, having resolved both concerns regarding the previous test.

B. Configuration Bitstream

Upsets in the configuration bitstream may result in erroneous processing. Many Virtex resources are left unused even, in very dense designs, so not every upset will result in incorrect processing. The Virtex SelectMAP interface allows configuration readback while the device is in use—a feature

we exploit to detect upsets. In addition, the Virtex can be partially configured, which speeds the recovery time.

The architecture of the configuration bitstream is shown in Figure 2. The Block SelectRAM (or BlockRAM) bits are on the outside edges of the device and are not included in a readback of the central, CLB section of the device, i.e., the CLB section includes all of the non-BlockRAM-related configuration bits. The XQVR1000 has a CLB section divided into 4778 individually accessible configuration frames, which run vertically. A frame is the smallest amount of data that can be read back or partially configured into the device.

Considering this, our bitstream upset detection and correction scheme does not track the exact location of a bitstream upset, just which frame contains incorrect data—each upset is treated with equal significance. Our bitstream upset management scheme [3][5][6] uses a controller to continually readback the configuration of the Virtex device and calculate a Cyclic-Redundancy Check (CRC) word for each frame in the bitstream. Each frame’s CRC word is compared to a CRC word that has been precomputed on the ground. An error indicates the presence of an upset in the frame. When an error is detected, the frame is reconfigured with the correct data. The continuous readback and CRC calculation take place in hardware in the configuration controller, relieving the system microprocessor from the task. Since readback is performed continually after initial configuration, the SelectMAP interface pins must be reserved for readback during normal operation and cannot be used as user IO.

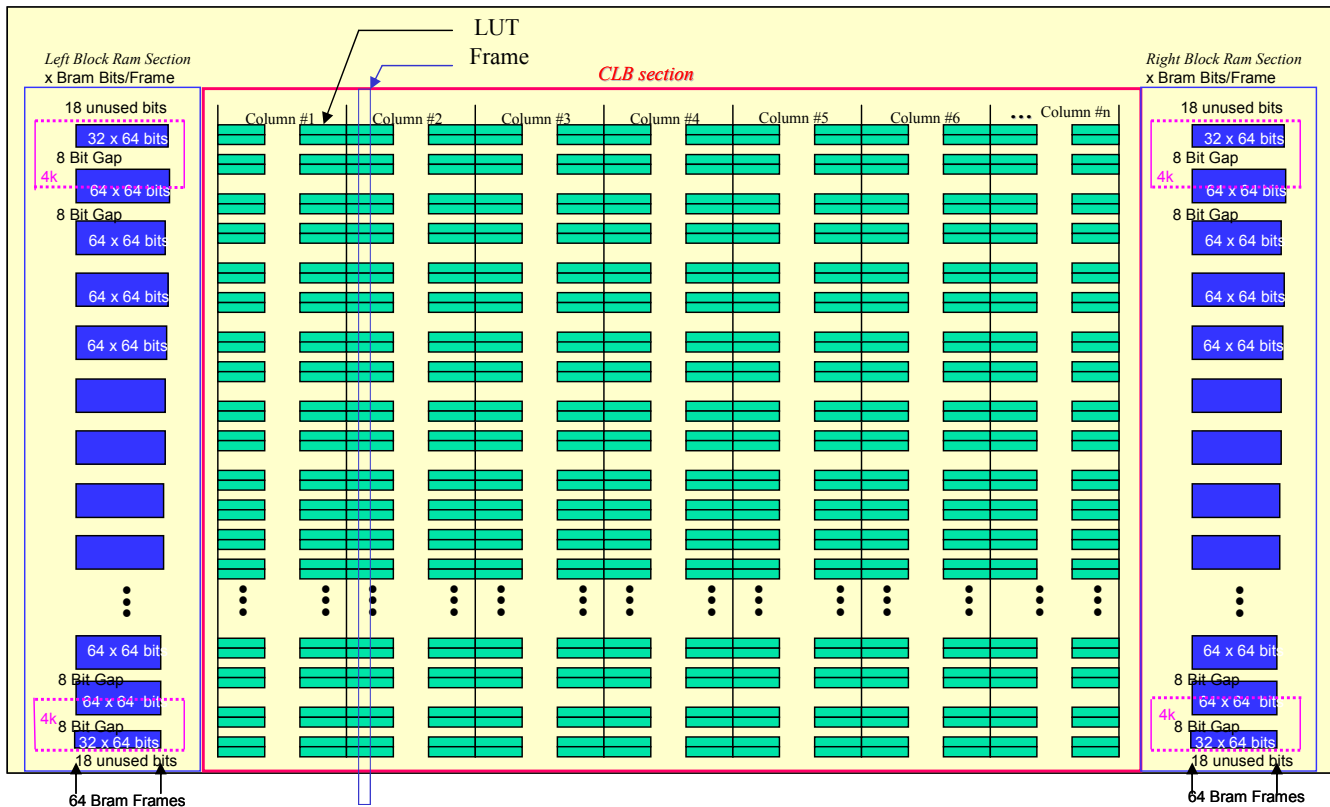
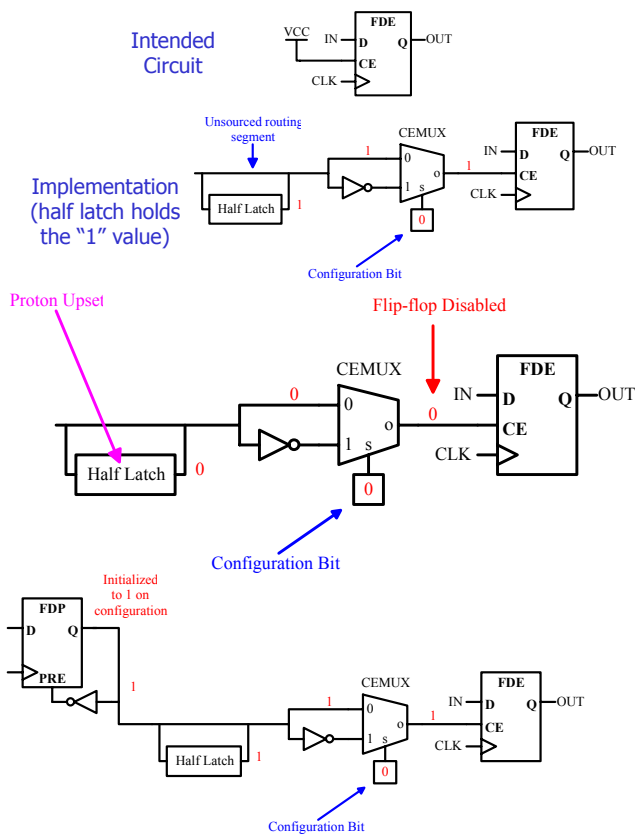


Figure 2: Architecture of Virtex configuration bitstream.



A flip-flop with appropriate feedback can be made to be resistant to SEUs in its state as shown above (PRE is an asynchronous set to "1").

Figure 3: Half-latch illustration showing intended circuit and half-latch immune implementation.

C. Block SelectRAM

Because the BlockRAM cannot be read safely via the SelectMAP interface while it is in use, upsets must be tolerated or mitigated using Error Control Coding (ECC) for detection and correction if desired; checksums, parity, or CRCs can be used for upset detection. The using BlockRAM is not materially different from using external SRAM sensitive to upset. Any mitigation must be implemented in logic with penalties in density and power.

D. Flip-Flops

Reducing the consequences of upsets in user flip-flops (both CLB and IOB) requires the use of redundancy. While the state could be observed using the 'capture' mode of readback, there is frequently no way to predict *a priori* a design's complete, correct state if it is performing a complex function. The impact of an upset in a flip-flop in our system varies with the application. In finite-impulse response (FIR) portions of the design, the errors will eventually be flushed out. In infinite-impulse response (IIR) portions, such as many finite state machine controllers and some signal processing hardware, the design may never recover on its own. This suggests that if we are to spend logic and power on reliability, we gain more by focusing on IIR structures. Our project intends to manage these on an algorithm design basis, so each algorithm design

can have a unique reliability versus throughput and power tradeoff.

E. Half-Latch structures

Another problem encountered when using Xilinx Virtex FPGAs in a radiation environment is that certain circuits, called half-latches, which generate many of the constant "0" and "1" values used by designs on Virtex FPGAs, are also susceptible to SEUs. When upset, the output values of these circuits will remain inverted until the device is fully reprogrammed. Further, this inversion is not directly observable through the configuration bitstream.

An example of the half-latch issue and a solution is shown in Figure 3. In this case, the unused clock enable input is driven by a half-latch. If the half-latch is upset, it will disable the flip-flop, modifying the intended function of the circuit. This modification cannot be observed through reading back and checking the configuration bitstream.

At an architectural level within Virtex FPGAs, half-latches drive IOB, slice, Block SelectRAM and other resource inputs when there are no direct sources for the input, i.e., when the inputs are left unconnected. Half-latches are a very efficient and ubiquitous source of "0" and "1" values throughout the device, no LUTs or other resources are required to generate constant logic values. Consequently, the Xilinx implementation tools generously use them throughout most designs.

The following resources or inputs can be driven by half-latches in the Virtex architecture:

Table 2: Virtex elements whose inputs can be driven by half-latches.

Structure	Inputs
BLOCKRAM	WEAMUX, ENAMUX, RSTAMUX, WEBMUX, ENBMUX, RSTBMUX
BSCAN	TDO1MUX, TDO2MUX
CAPTURE	CAPMUX
DLL	RSTMUX
GCLK	CEMUX
IOB/PCIIOB	SRMUX, TRIMUX, TCEMUX, OMUX, OCEMUX, ICEMUX
PCIOLOGIC	I1MUX, I2MUX
SLICE	BYMUX, BXMUX, CEMUX, SRMUX, F1-F4, G1-G4
STARTUP	GWEMUX, GTSMUX, GSRMUX
TBUF	TMUX, IMUX

Within the FPGA Editor tool provided by Xilinx, the use of a half-latch is expressed as a constant "0" or "1" input into the above mentioned muxes. In reality, these muxes only have the ability to select their inputs or inverted versions of their inputs and the constants illustrated in FPGA Editor are values produced by half-latches.

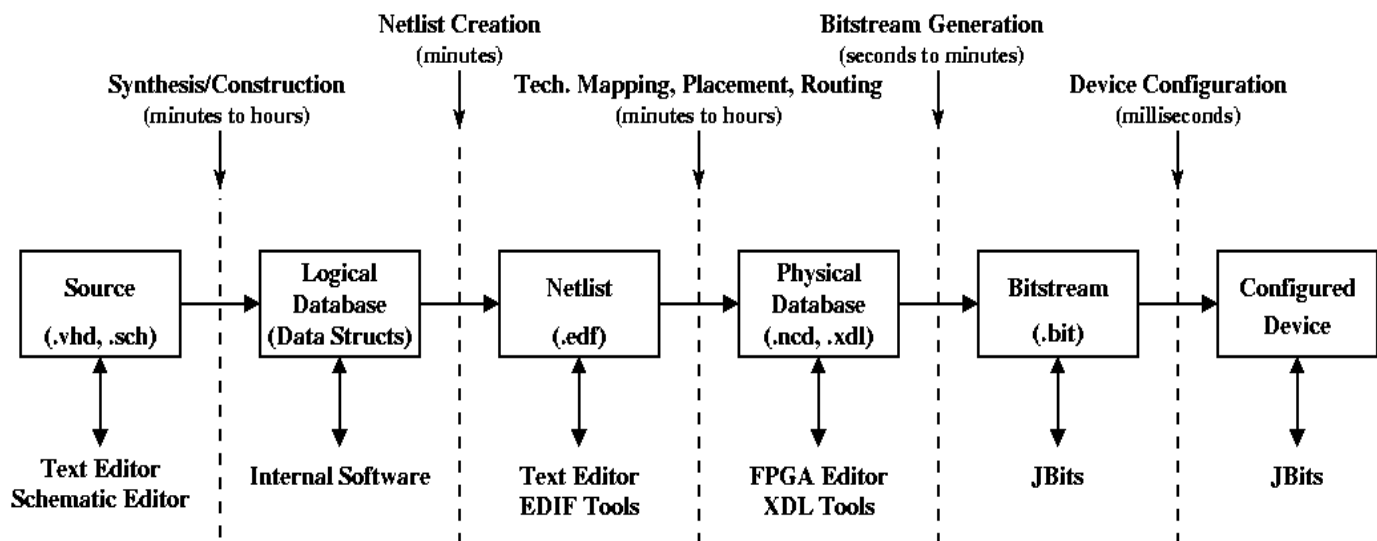


Figure 4: Xilinx FPGA Design Flow and Design Forms

In general, the half-latches driving the inputs to the above mentioned muxes are the critical half-latches. Modification to the values at the inputs of these muxes can have serious consequences to the operation of circuits. On the other hand, the unused inputs to the LUTs (F1-F4, G1-G4) are not as critical since the logic functions are encoded redundantly within the LUT such that “0” or “1” values on the “don’t” care inputs result in the same output value.

The solution to the hidden half-latch inversion problem is to remove the reliance on half-latches that produce the constant “0” and “1” values in designs by routing an explicit “0” or “1” value to these mux inputs. The sources of these explicit “0” and “1” values should be created by using resources which can be explicitly configured with the bitstream so that SEU errors which cause changes to these constant values can be handled via bitstream error detection and correction methods. Some potential sources for constant values include FPGA input pins driven externally by a logic “0” or “1”, LUTs filled with an all “0” or all “1” value (as appropriate), or flip-flops as shown in Figure 3. Note that the flip-flop circuits self-correct if the flip-flop state is upset via radiation. Other constant sources are likely possible. Again the important feature is that if these sources of constant logic values are disturbed by SEUs, the error in their configuration can be detected using bitstream readback techniques—these sources of constant logic values can still be affected by bitstream SEUs.

The half-latch removal process can be performed at several different stages in the design flow, from the HDL or schematic-entry level down to the bitstream level, as Figure 4 shows. As a general comment, the level of design abstraction decreases as a design moves from left to right in the figure.

Approaches that modify designs earlier in the process are less likely to eliminate all half-latches since synthesis, technology mapping, and, possibly, placement and routing may introduce half-latches into the design implementation. Careful design may insure that half-latches are never used in

the design in the first place, but this will require fairly involved design practices, such as requiring that the explicitly generated constant “0” and “1” values be connected throughout the design and that HDL descriptions be carefully structured so that half-latches are not introduced. As an additional complication to these approaches, design practices which worked for older synthesis and technology-mapping tools may not work for new tools as synthesis and technology mapping techniques evolve. If the use of constants and, thus, half-latches were controllable in the synthesis and technology mapping stages via a switch or parameter, most, if not, all of the half-latch problems would not exist.

The *Physical Database* and the *Bitstream* representations of the design are probably the more promising design representations to modify since they can or do represent the final placed-and-routed forms of the designs. So, if the half-latches are removed at these points in the design flow, half-latch problems should not exist. With these latter design representations, the designer is dealing with the design expressed in terms of FPGA resources at a very low level of abstraction.

Modifying designs while in these low-level forms does have a few disadvantages. First, only a handful of tools are useful for manipulating designs at this level. As referenced in Figure 4, FPGA Editor from Xilinx can be used to modify the design in the form of a Native Generic Database File (.ncd). Further, Xilinx provides a program called *xdl* that converts the proprietary Native Generic Database File format into a published, open format called the Xilinx Design Language (XDL). Once converted to XDL, third-party tools can be written to manipulate the XDL and the modified design can be converted back to the Native Generic Database format for bitstream creation. JBits is also a possibility for modifying designs at the bitstream level, but currently less than 100% of Virtex and Virtex-E devices are configurable via JBits (as of version 2.8). As a second disadvantage, because of the low abstraction level, a detailed knowledge of the devices is

required to make the modifications or design tools to automate modifications. Finally, because designs are placed and routed when they are being modified, there is a slight possibility that routing or logic resources may be too limited to remove the half-latches.

Currently, we have created a tool which parses the XDL representation of a design to locate half-latch issues and then generates a script for FPGA Editor to automate the removal of all half-latches. The initial version of the tool uses a single FPGA pin to generate a logic “1” and that source is routed to all muxes initially having half-latch sources. The muxes are configured as inverting (to produce a “0”) or non-inverting (to produce a “1”) depending on the constant value required. We will be creating other versions that use other strategies for removing half-latches and intend to test these half-latch removal techniques under proton radiation to understand which techniques are best.

F. Single Event Functional Interrupts

Single event functional interrupts (SEFIs) are those SEUs that have unusually far reaching consequences. The SEFIs we believe exist in the Virtex include:

1. JTAG TAP controller
2. Configuration control state machine reset (FSM POR)
3. SelectMAP configuration pin upsets

The total cross-section for these three signatures is measured less than $1E-5 \text{ cm}^2$, as mentioned previously and all can be detected by the unusually large number of ‘upsets’ detected during a device readback. Each of these SEFIs interferes with device readback and results in reading many more errors than actually exist.

The JTAG TAP controller in the Virtex device does not make the reset pin available to the user to hold the controller in reset while deployed. It has been reported [4] that JTAG TAP controllers can upset, possibly moving the device into an undesirable state. The approach to managing this when reset is unavailable is the same for the Virtex as for other devices: place a pull-up resistor on the mode pin and wire a free running clock to the test clock input. In the event of an upset the controller will return to the reset state in 5 clock cycles. It is important that the test clock input is not shared with any other pins on the Virtex to prevent contention on the clock in the presence of an upset, thereby preventing recovery. Though we have not observed a TAP controller upset in accelerator experiments on the Virtex, the possibility must exist as with any device using a JTAG implementation which leaves test reset inaccessible. The cross-section must be small in relation to dominant upset signatures.

The configuration management circuit has also has a sensitive cross-section. When upset the device behaves as though PROGRAMn has been asserted, the configuration is cleared. Because of the infrequent nature of this upset mode we do not have a mitigation plan for it except to reconfigure and begin processing again with discarded data.

The configuration of the device pins used for the SelectMAP interface (D0-7, WRITEn, CSn, BUSYn, etc) may also be upset. This results in an inability to read or write configuration data to the Virtex. It can be detected when

reading back bitstream data because a significant percentage of the bitstream will be wrong. Asserting PROGRAMn and then reconfiguring the device can correct the problem.

III. SYSTEM DESIGN

A. SEU simulation

An in-system SEU simulation/injection capability is extraordinarily useful in verifying the upset detection portion of a system. Partial configuration can be used to inject frames with single-bit errors into the Virtex device so the system’s behavior can be observed to understand upset consequences. This is most conveniently implemented with microprocessor access to the device SelectMAP interface and a source copy of the bitstream.

For our work, we have also used a PC-based SEU simulator to test the half-latch elimination scheme and to identify other bitstream-related SEU sensitivities, such as the sensitivity of the SelectMAP interface. The simulator, a USC/ISI SLAAC1-V board [10] with custom software, loads the same design into two Virtex FPGAs and injects single-bit errors into one of the device’s configuration bitstream through partial configuration. A third FPGA is used to do a real-time comparison of the two FPGAs’ outputs to identify when a design’s function has been changed due to a simulated bitstream SEU. In the future, this simulator will be used to establish the usefulness of further SEU mitigation schemes.

For SRAM FPGA systems exposed to radiation, the possibility of internal contention that is initiated by a corrupted programming bitstream and causes permanent damage in the device is a concern. In our studies we have upset millions of bits, one at a time, and never observed a permanent fault. In addition, during static SEU testing a significant percentage of the bitstream was upset (25% or more), with the accumulated upsets resulting in larger (an increase of more than 1 Amp) standby current consumption. This condition persisted for a few minutes and was repeated ~10 times per device on several different devices. No permanent damage has ever been detected in our experiments; however, the devices in question have never been retested by the factory to guarantee full compliance with specification. A concept for silicon testing the Virtex with a portfolio of configurations that exercise different elements of the architecture has been presented [8]; it has intriguing implications for assurance of deployed systems. In addition, it is possible that fault identification would allow designing around a defect, perhaps squeezing more life out of a degraded system.

B. Device IO Upset Implications

One important consideration for our system that SEU simulation revealed is that SEUs in the configuration data for bi-directional, tristatable IO pins can cause contention. Our upset controller and the three Virtex FPGAs all share a local bus that the payload controller uses to interface to the module. When the SEU controller detects an error in the bitstream, it generates an interrupt to the payload controller, which then retrieves status information and clears the interrupt via register accesses over the local bus. When a Virtex device

inadvertently drives this bus, the payload controller cannot repair the bitstream nor can it clear the interrupt. We manage this event by resetting the entire reconfigurable module, which clears the Virtex configurations and releases the bus. We have not observed any permanent faults in the Virtex or other components of the system in our simulations that result in this contention, but it is inadvisable to let the condition persist. We have not observed inputs being turned into outputs as a result of a single configuration upset. Exhaustive testing suggests single bit errors cannot have this effect [9].

C. Readback and Configuration Considerations

There are a few details to consider when performing readback on Virtex devices. First, do not perform readback on designs that use SelectRAM primitives such as SRL16. The combination of using LUTs as RAM and performing readback interferes with design operation and, potentially, device configuration—the two are incompatible.

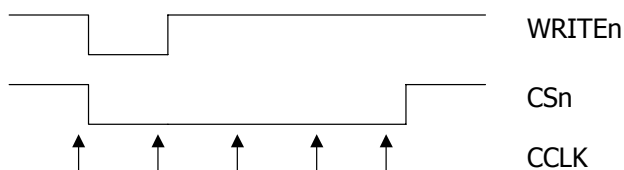


Figure 5: Abort sequence is performed prior to any command sequence issued on the Virtex SelectMAP configuration interface.

Second, always perform an abort sequence before issuing commands to the device, as shown in Figure 5. That will insure that the FSM is in a known state each time commands are issued. When issuing commands, always start with a dummy and sync word before sending configuration commands, as shown in Table 3. This insures that the configuration FSM is resynchronized within the Virtex device. Synchronizing in this fashion guarantees that the FSM correctly assembles the bytes on the SelectMAP interface into the proper 32-bit command words.

Table 3: Example readback command sequence for a full readback of a 1000K device. Note the inclusion of a dummy word and synchronization word.

EXAMPLE READBACK COMMAND SEQUENCE

```
x"FF",x"FF",x"FF",x"FF", -- dummy word
x"AA",x"99",x"55",x"66", -- sync word
x"30",x"00",x"20",x"01", -- write to FAR
x"00",x"00",x"00",x"00", -- data: start frame address
x"30",x"00",x"80",x"01", -- write to CMD
x"00",x"00",x"00",x"04", -- data: RCFG
x"28",x"00",x"60",x"00", -- read from FDRO
x"48",x"02",x"D8",x"0D" -- data: # of data words
```

When commanding the Virtex for a readback, always read back the amount of data that was requested in the command sequence before issuing new commands. Otherwise, the internal counter providing the readback data will not reset, leaving the Virtex readback logic in a confused state. Recovery from this confused state is possible by performing a partial configuration of a frame after an aborted readback—this will reset the configuration controller’s internal counter. This recovery scheme may be useful when an SEU is detected

midway through the readback of a device; in this case, the system can abort the readback and reconfigure the offending frame.

Pad data is required when partially configuring Virtex FPGAs. It is important that “0” values be used for pad data because the pipeline registers return some pad data in the readback stream. Using nonzero pad data may result in false detection of SEUs in the readback bitstream if pad data is not carefully masked out of the CRC calculation.

IV. CONCLUSION

This paper has described various single-event upset categories for the Virtex FPGA, each category having unique observability properties. Bitstream upsets have excellent observability and simple mitigation, while half-latch upsets are not observable and require user design modifications for mitigation. A methodology for eliminating half-latch sensitivities has also been described. Additionally, known SEFI signatures are described along with suggestions for identifying and recovering from them. Finally, several practical design considerations have been presented which should help others avoid various pitfalls relating to SEU mitigation and Virtex devices. We believe that we can successfully use Virtex FPGAs for processing our remote sensing data on orbit.

Future work includes proton testing to validate the performance of our half-latch removal concepts, to verify the accuracy of the PC-based SEU simulator, and to take another look at the upset categories to be sure that all have been accounted for. Determining the presence of transient upsets, or their absence, is also an objective of our proton testing.

V. ACKNOWLEDGMENTS

This work was funded by the United States Department of Energy.

REFERENCES

- [1] E. Fuller, et al, “Radiation test results of the Virtex FPGA and ZBT SRAM for Space-Based Reconfigurable Computing,” MAPLD 1999 Proceedings, C2.
- [2] C. Carmichael, “Triple Module Redundancy Design Techniques for Virtex FPGAs”, www.xilinx.com, XAPP197, November, 1, 2001.
- [3] E. Fuller, et al, “Radiation Testing Update, SEU Mitigation, and Availability Analysis of the Virtex FPGA for Space Reconfigurable Computing,” MAPLD 2000 Proceedings, P30.
- [4] R. Katz, et al, “USING IEEE 1149.1 JTAG CIRCUITRY IN ACTEL SX DEVICES,” www.actel.com, August 25, 1998.
- [5] C. Carmichael, “Correcting Single-Event Upsets through Virtex Partial Reconfiguration”, Xilinx Application Note XAPP216, June, 2000.
- [6] C. Carmichael, et al, “Proton Testing of SEU Mitigation Methods for the Virtex FPGA”, MAPLD 2001 proceedings, P6.
- [7] F. Lima, et al, “A Fault Injection Analysis of Virtex FPGA TMR Design Methodology”, RADECS, September 2001 Proceedings.
- [8] P. Sundararajan, “Testing FPGA Devices using JBits”, MAPLD 2001 proceedings, E5.
- [9] N. Rollins, et al, “Reliability of Programmable Input/Output Pins in the Presence of Configuration Upsets,” MAPLD 2002 proceedings, C3.
- [10] B. Schott, et al, “Reconfigurable architectures for system level applications of adaptive computing,” VLSI Design vol. 10, no. 3, p. 265-79, 2000.