

LA-UR-02-3005

*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

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

¹Los Alamos National Laboratory, Los Alamos NM, USA

²Brigham Young University, Provo UT, USA

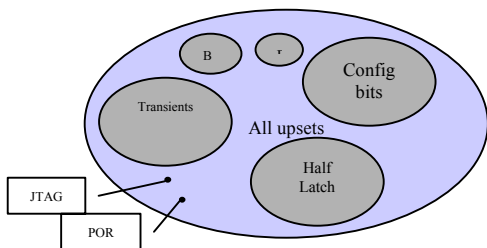
Abstract—Field Programmable Gate Arrays (FPGAs) are indisputably useful for space missions where system schedule and cost are critical but production quantity is low. SRAM based FPGAs are uniquely suited for remote missions because of the ability to change function *in situ* and because they offer substantial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

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

I. INTRODUCTION

We intend to use the Virtex SRAM based FPGA in an orbital ionospheric study. 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 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. Our goal is to make as many of the FPGA resources available to the processing algorithm as possible and still achieve the reliability required by our system. The consequences of upsets will have variable severity depending on where in the device they occur. The paper describes what we know about device-upset categories and what techniques at the device and system level that can be used to increase observability or mitigate risk. Unresolved questions are also mentioned.

For our system we wish to preserve the logic for our application and use little for redundancy. Upsets are acceptable, but we want them to be observable if possible. Data upsets (shot noise) are more acceptable than state machine upsets that could result in incorrect processing indefinitely.



II. SENSITIVE CROSS-SECTION

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

We have an approach to managing some of the unique upset signatures found in the Virtex. Xilinx [?] describes techniques using redundancy that further enhance performance, of course with a density and power tradeoff. Our approach emphasizes detection of upsets in the configuration bitstream and repairing them via partial reconfiguration of the FPGA to decrease recovery time. This requires that data or data product acquired during the SEU and recovery must be discarded. Using this approach reserves more power and logic for signal processing at the cost of availability. Depending on the application, there are various strategies for detecting upsets that occur outside the bitstream.

A. Transient Effects

The existence of transient induced upsets has not been established, but we know [MAPLD2000 ref] from proton accelerator experiments in 2000 that 45% of upsets detected during dynamic testing were not due to configuration bitstream upsets.

B. Configuration Bitstream

Upsets in the configuration bitstream may result in erroneous processing. Even in very dense designs, many Virtex resources are left unused. This means that not every upset will result in incorrect processing. Our system does not categorize or track the consequence of a bitstream upset. Each upset is treated with equal significance. In the future, system availability could be extended if a tracking scheme can be developed to categorize the importance of each bit in the configuration.

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 configuration bitstream is divided into approximately 4800 frames for the XCV1000; a frame is the smallest amount of bitstream data that can be updated.

The only constraint on the readback operation is that the LUTs cannot be used as distributed RAM elements like the

* This work is supported by US Department of Energy

SRL16. The use of these memory elements causes problems in the readback bitstream. Some of our applications require these constructs (they are very useful distributed delay elements for DSP), so those applications must have upset detection built into the design logic. For example, Ray Andracka has designed an FFT engine that uses down time between vector processing to generate a LFSR signature and run it through the FFT. Evaluation of the output is used to indicate the presence of a persistent upset in the bitstream.

Our bitstream upset management scheme uses a controller to readback the configuration of three Virtex devices sequentially when enabled. Initially, the payload controller, via a software interface, configures each Virtex. Each bitstream is accompanied by a 'CRC codebook' that is stored in the upset controller's local SRAM by the payload controller. After the Virtex FPGAs are configured, the upset controller is enabled and it begins to read each frame of the devices' configuration; it generates a new CRC for each frame and compares it to the codebook CRC. An error indicates the presence of an upset in the frame. The continuous readback and CRC calculation take place in hardware in the controller, relieving the system from the task.

When an error is detected, the upset controller generates an interrupt to the payload controller (microprocessor) that then disables readback and partially configures the device with the new frame using software. The payload controller notes the device, frame, and time of the upset for state of health purposes and resets the Virtex. Readback can be re-enabled once the frame has been repaired.

In our system, a readback is executed on each of 3 Virtex every 180 ms. When an upset is detected, the system returns to service in ?? ms. The worst case forecast for upset rate in our payload with nine XCV1000 devices is about once every ten minutes (polar regions and the South Atlantic Anomaly), so we do not foresee an undue processing burden on the payload controller. Our system recovery time is a strong function of the format of the source bitstream. It can be stored in payload controller SRAM or in system FLASH memory in either compressed or uncompressed format. Recovering frame data from a compressed bitstream in FLASH requires the most recovery time with the benefit of leaving more SRAM available to the payload controller and storing more configurations in the system. The scheme can be modified on orbit to increase availability or to accommodate more processing modes.

When the upset controller commands the Virtex to provide readback data, we always initiate an abort sequence followed by a dummy word and synchronization word in the command sequence. This assures the internal Virtex state machine is in the expected state. The command sequence always requests the entire CLB configuration data. It is important that all data be read from the Virtex after each command. Failure to complete the readback leaves the state machine inside the Virtex in a mode that tries to supply all the data. Giving another readback command will result in unknown data being supplied by the device. When our upset controller detects an upset, it then completes the readback of the device before partial configuration is initiated. The internal state machine of the Virtex can be reset by a partial configuration, but we

prefer the regularity of simply completing all readback operations.

C. Block Ram

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

D. User Registers

Reducing 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 in readback, there is no way to predict the correct state if the design is performing a useful 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 effect will be flushed. In Infinite Impulse Response systems (IIR) such as many finite state machine controllers and some signal processing 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, making it hard to know whether or not a design is functioning normally based on tests that validate the FPGA devices' programming bitstreams.

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. This results in a very efficient and ubiquitous source of "0" and "1" values throughout the device which do not require LUTs or other resources to be used to generate constant logic values. Consequently, the Xilinx implementation tools generously use half latches throughout most designs.

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

- ## SLICE: BXMUX, BYMUX, CEMUX, SRMUX, F1-F4, G1-G4 (12 total)
- ## IOB: SRMUX, TRIMUX, TCEMUX, OMUX, OCEMUX, ICEMUX (6 total)
- ## BRAM: WEAMUX, ENAMUX, RSTAMUX, WEBMUX, ENBMUX, RSTBMUX (6 total)
- ## GCLK: CEMUX (1 total)

DLL: RSTMUX (1 total)

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.

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 1. Note that the flip-flop circuits self-correct if the flip-flop state is upset via radiation. Other 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.

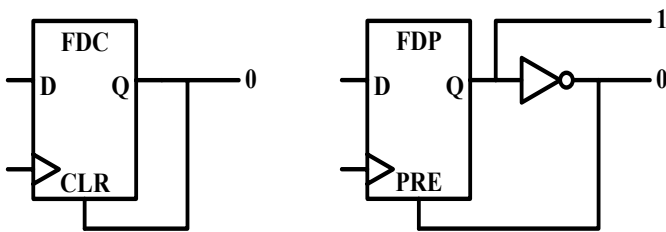


Figure 1 Self-correcting Flip-Flop Sources for Constant Logic Values

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 2 shows. As a general comment, the level of design abstraction becomes lower as a design moves from left to right in the figure.

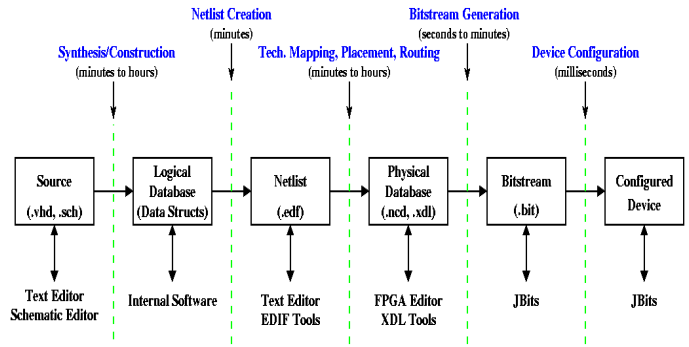


Figure 2 Xilinx FPGA Design Flow and Design Forms

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 and, thus, these design forms represent the design 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 2, 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" which converts the proprietary Native Generic Database File format into a published, open format called the Xilinx Design Language. 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. Of course, 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 are required to make the modifications or design tools to automatically make the modifications. Finally, because designs are placed and routed when they are being

modified, there is a slight possibility that 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 better than others.

F. Jtag TAP controller

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 very small, resulting in a very low frequency of occurrence in orbital applications (some number of years). 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 [Katz et al ref] that the controller can upset, putting the device into many undesirable states. 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 should an upset occur. In the presence of contention there is no guarantee that the system will recover.

G. POWER on reset in Configuration controller

The configuration management circuit has also has a sensitive cross-section. When upset (years or more frequency for orbital application) the device behaves as though PROGRAM 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.

III. SYSTEM DESIGN

A. Device IO & Upset Implications

We experiment with upsets in our system by using the partial reconfiguration feature to inject configuration frames with errors to simulate upsets. We can then evaluate the performance of the SEU management system as well as the consequence of bitstream upsets. One important consideration for our system that this testing uncovered is the consequence of upsets in bi-directional, tristatable I/O. 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 that then reads status across the local bus. The interrupt is also cleared via a register

access on the same local bus. When the Virtex 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 or accelerator experiments.

IV. CONCLUSION

We are still working to understand the upset signatures that are possible in the Virtex. In our SEU simulation experiments we have observed that partial configuration may not entirely repair the state of the Virtex. The experiments suggest that partial configuration does update the bitstream to the expected value and return the circuit to correct operation. However, when upsetting subsequent bits in the bitstream there seems to be increased sensitivity (a given bit is more likely to cause an error) as testing continues. When experimenting with a full and complete device configuration after each simulated upset, we do not observe these effects. Further work is needed to understand the signature and possible mitigation.

We are also planning to further characterize the effects of mitigating the half latch sensitivity. We intend to find the solution that offers the smallest increase in sensitivity while eliminating the unobservable upset mode. It is important to have an automated tool flow implementation of the half latch removal so there are no constraints on the logic designer.

The ability to inject simulated SEUs into the bitstream of the device had been an invaluable aid to understanding the device and system level behavior. We believe our payload experiment will be a substantial leap forward in performance. We expect the system to handle the various categories of upsets discussed in this paper and that it is reasonably comprehensive list. The goal is to have few unpleasant surprises while on orbit.

V. ACKNOWLEDGMENTS

I would like to thank Mark Dunham, Manuel Echave, Charles Fite, Anthony Salazar, and many others at Los Alamos National Laboratory for their hard work and commitment as well as Rick Padovani, Joe Fabula, and Carl Carmichael of Xilinx. I also thank Randy Bell at the Department of Energy for his support of this project.

REFERENCES

- [1] E. Fuller, et al, "Radiation Testing Update, SEU Mitigation, and Availability Analysis of the Virtex FPGA for Space Reconfigurable Computing," MAPLD 2000 Proceedings, P30.
- [2] C. Carmichael, "Correcting Single-Event Upsets through Virtex Partial Reconfiguration", Xilinx Application Note XAPP216, June, 2000.
- [3] C. Carmichael, et al, "Proton Testing of SEU Mitigation Methods for the Virtex FPGA", MAPLD 2001 proceedings, P6.
- [4] F. Lima, et al, "A Fault Injection Analysis of Virtex FPGA TMR Design Methodology", RADECS, September 2001 Proceedings.
- [5] P. Sundararajan, "Testing FPGA Devices using JBits", MAPLD 2001 proceedings, E5.