LA-UR- 09-04566

Approved for public release;
distribution is unlimited.

Title:	Reliability Concerns with Logical Constants in Xilinx FPGA Designs
Author(s):	Heather Quinn, Paul Graham, Keith Morgan, Patrick Ostler Los Alamos National Laboratory Greg Allen -- The Jet Propulsion Laboratory Gary Swift, Chen Wei Tseng -- The Xilinx Corporation
Intended for:	IEEE NSREC 2009 Quebec City, Quebec, Canada

Los Alamos
NATIONAL LABORATORY
EST. 1943

Form 836 (7/06)

Reliability Concerns with Logical Constants in Xilinx FPGA Designs

Heather Quinn, Greg Allen, Gary Swift, Chen Wei Tseng, Paul Graham, Keith Morgan, and Patrick Ostler

Abstract— **In Xilinx Field Programmable Gate Arrays logical constants, which ground unused inputs and provide constants for designs, are implemented in SEU-susceptible logic. In the past, these logical constants have been shown to cause the user circuit to output bad data and were not resetable through off-line reconfiguration. In the more recent devices, logical constants are less problematic, though mitigation should still be considered for high reliability applications.**

I. Introduction

Recently, the Xilinx Virtex family of reconfigurable field-programmable gate arrays (FPGAs) have gone through radiation experiments to qualify these devices for space. While these devices could provide a cost-effective, flexible platform for space-based data processing, the devices are susceptible to single-event upsets (SEUs). Unlike application-specific integrated circuits (ASICs), the user's circuit is stored in static random access memory, which means both the circuit and the circuit state could be altered by SEUs. In this paper, we will focus on a specific failure mechanism caused by SEUs in the logical constants.

Logical constants are needed to generate zero and one logic values and are an artifact of mapping VHDL/Verilog designs to the FPGA architectural elements. Half latches were previously discussed in [1]. That paper presented data that showed that the Virtex-I half latches had a number of problems. First of all, once upset the upset half latch would persist the incorrect value. Second, the half latches could not be "scrubbed" through on-line reconfiguration, and only returned to the configured value through full off-line reconfiguration of the device. As off-line reconfiguration interrupts the circuit's operation, this method of removing SEUs from the device is not desirable. Furthermore, a design's half latches could not be directly observed. The only indication that a half latch was upset was through changes in output data. While this observation is easy at the accelerator when the output is being compared to golden output data, this type of observation is not as easy on orbit where the output data might not be known. For these reasons, half latch mitigation was easier than repairing and monitoring half latches. While most of these problems remain true in the later devices, starting in the Virtex-II the half latches would return to their configured value after some time.

There are two types of logical constants that are used in the Virtex family devices: implicit and explicit. Implicit logical constants are often used to "tie off" unused inputs to input/outputs, memory, clocking, user flip flops, and other resources to known values, as shown in Figure 1(a). Implicit logical constants are implemented as half latches (weak keeper circuits) for all of the Virtex family devices. When enabled, the weak keeper pulls a floating input to a known zero or one level.

The explicit logical constants are often used to tie off the zeroth bit of carry chains for adders (as shown in Figure 1(b)), tie off unused inputs to the embedded multipliers/DSP cores and any user-specified constants in the design. In the Virtex-I and Virtex-II explicit logical constants are implemented with *constant LUTs*, where the LUT's configured equation is a one or zero value. In the Virtex-4, explicit logical constants have two possible implementations — an architectural post that provide access to ground rails for each configuration logic block (CLB) and constant LUTs.

As both the half latches and constant LUTs use SEU-susceptible logic or memory, these constants are affected by radiation. Furthermore, because they often affect inputs to user flip flops, adders, and multipliers,

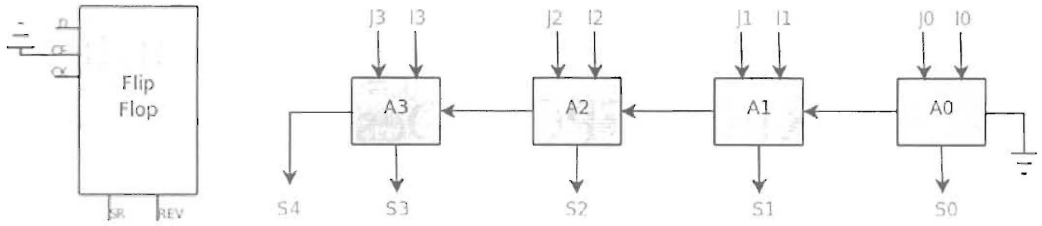
Document release number: LA-UR-09-00763. This work was funded by the Department of Energy through the Deployable Adaptive Processing Systems and Sensor-Oriented Processing and Networking projects, NASA through the Reconfigurable Hardware in Orbit project under AIST contract #NAG5-13516, and the Air Force Research Laboratory under the FPGA Mission Assurance Center.

H. Quinn, P. Graham and K. Morgan, are with ISR-3 Space Data Systems, Los Alamos National Laboratory, Los Alamos, NM, 87545 USA (e-mail: hquinn@lanl.gov)

P. Ostler is with Brigham Young University, Provo, UT, 84602 USA

G. Allen is with Jet Propulsion Laboratory, 4800 Oak Grove Dr, Pasadena, CA USA

G. Swift and C. Tseng are with Xilinx Corporation, 2100 Logic Drive, San Jose, CA, USA



(a) Clock Enable Tied to Vcc

(b) Zeroth Bit of the Carry In Chain for the Adder Tied to Ground

Fig. 1. Different Types of Logical Constants

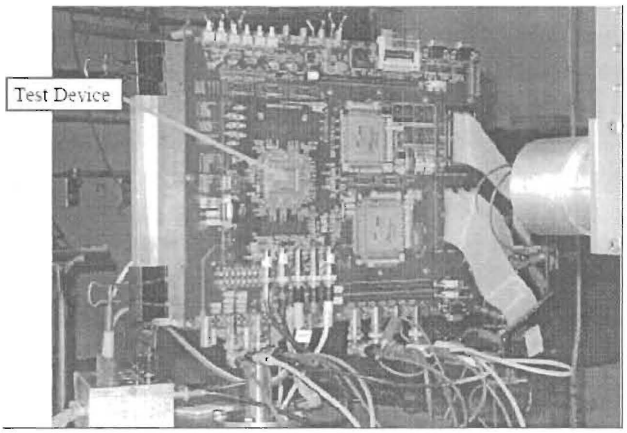
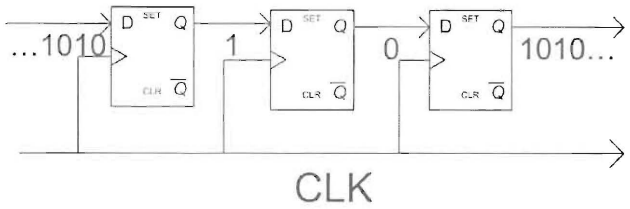
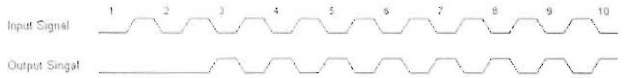


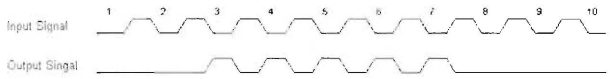
Fig. 2. Test Fixture at Texas A&M Cyclotron



(a) Example of the Shift Register Using Three User Flip Flops



(b) Example Input and Output Data Signals When Operating Correctly



(c) Example Input and Output Data Signals When Malfunctioning

Fig. 3. Example Design and Signals for the Shift Register Design

their failure can lead to the corruption of intermediate processing values and output data. Fortunately, their cross-section and the number of half latches are smaller than that of the configuration memory, making them less problematic than the configuration memory.

In this paper, we will present the reliability concerns of both the implicit and explicit logical constants in Section II, including previously unpublished radiation test data for the Virtex-II and Virtex-4 devices. In Section III we will discuss options for mitigating logical constants, including a new method for mitigating constant LUTs.

II. Radiation Test Data and Analysis

In this section we will provide an analysis of the reliability concerns with the three types of logical constants. We provide radiation test data for half latches on the Virtex-II and the Virtex-4 devices.

A. *Implicit Logical Constants*

The Virtex-II, and Virtex-4 devices were tested at Texas A&M and Lawrence Berkeley National Laboratory cyclotrons. They were tested to determine the cross-section of the half latches. Because the half latches return to their normal configured value, we also analyzed the data for how long the upset value would persist, called the *hold time*. Table I lists the facility, kinetic energy, and species used to test each device. All devices were tested with nominal voltages and temperatures using the SEAKR test fixture, shown in Figure 2. This board includes two FPGAs that monitor the configuration of the FPGA (*configuration monitor*) and the output behavior of the design under test FPGA (*functional monitor*). Each device was tested with the same basic design of 32 shift register chains. The shift registers are built from user flip flops and not the SRL16 shift registers, as shown in Figure 3(a). When this type of latch is used, the tools will infer half latches on the clock enable and the set/reset signals to tie those inputs to known values. The length of the shift registers is customized for each device so that the device is completely utilized. As each user flip-flop in the device has at least two half latches, the 32 shift registers enables thousands of half latches to be tested, as shown in Table I.

To operate this design strings of alternating ones and zeros are input to each of the 32 shift registers. For example, as shown in Figure 3(b) the output data shows that the input signal is delayed by three clock cycles and that the output value is not stuck at either one

TABLE I
XILINX PARTS TESTED

Device	Facility	Kinetic Energy	Species	Half Latches per Design
XC2V6000	TAMU	25 MeV/u	Ne, Ar, Kr, Xe	134,528
XC4VSX55	TAMU	15 MeV/u	Ar, Ta	165,700
XC4VSX55	TAMU	25 MeV/u	Ar, Ne	165,700
XC4VSX55	TAMU	40 MeV/u	Kr	165,700

or zero. For this case, the shift register is operating properly. Under error conditions the output could be stuck at zero or one, as shown in Figure 3(c). Therefore, output errors are easy to detect through glitches in the output data. When an output error is detected in a shift register chain, a counter will increment each clock cycle until the error clears. These counters are attached to output pins and are sampled on average every 0.125 to 0.5 milliseconds to determine the current value of the counters. After a shift register returns to its configured value, the counter's final value persists for a few samples to ensure logging the value before being reset. The functional monitor on the SEAKR board logs the output of these errors and determines what the cause of the error is. Further analysis of the functional monitoring using custom software determines the histogram of hold times for all errors causing output errors and half latches.

To determine the cause of the output failure, the functional monitor must distinguish from a number of different causes for the erroneous output data. For this design, SEUs could cause these problems:

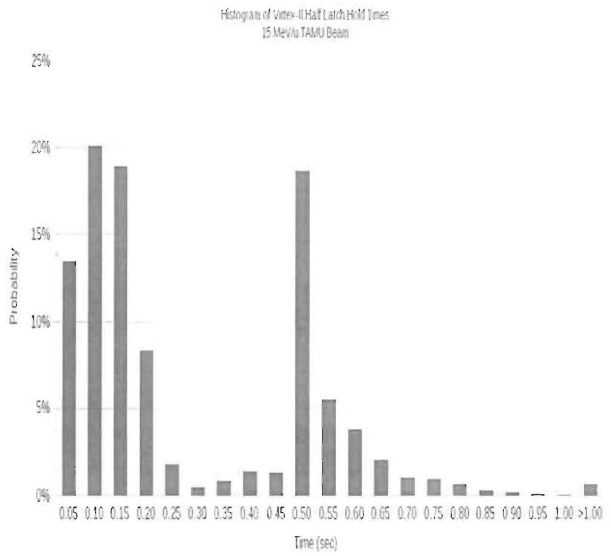
1) A change in a single user flip-flop,
2) A change in the configuration memory that causes the routing or the configuration of a user flip-flop to change, or
3) A change in the half latch that causes the user flip-flop to either be held in reset or to disable the clock.

It is necessary to detect which one of these changes occurred to cause the output error. For the first case, a direct strike will only cause a single output value to change for one clock cycle. While the upset user flip-flop will be overwritten in the next clock cycle, the bad data will continue through the shift register until it is output and thereby flushed from the system.

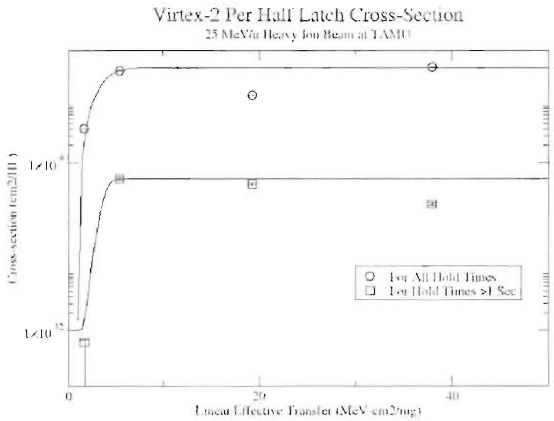
The second and third cases are harder to distinguish, as the effect on the output data is very similar. In these cases, the error will persist until fixed. Once the cause of the error is removed, the bad data will flush out of the system. Because the circuit is feed forward, errors to the configuration memory will only persist as long as it takes for the scrubber to fix the configuration memory, instead of through resetting the circuit. If the scrubber *definitively* fixes the bit and the bad output persists, then the cause of the bad output data is a half latch. Because it is not possible to determine the timing of the upset occurring and where the scrubber is its reconfiguration cycle, errors are not categorized as half latches until the error persists through two scrub cycles. One weakness of this approach is that half latches that returned to their configured value within two scrub cycles will not be recorded as half latches, causing possibly a higher count of errors in the affected time periods.

As shown in Figure 4(a), in the Virtex-II the distribution of hold times are bi-modal with a peak at 0.13 seconds and an impulse function at 0.53 seconds. On average, 99.4% of all half latches returned to their configured value within one second and 0.6% of all half latches did not return to their configured value after one second. The per-half latch cross-section is shown in Figure 4(b). This data shows that the maximum per-half latch cross-section is $4.98E-8 \pm 2.34E-10$ $\frac{cm^2}{HL}$ for all half latches and is $5.06E-10 \pm 4.11E-11$ $\frac{cm^2}{HL}$ for half latches that persist for longer than one second. When this data is compared to the bit cross-section for configuration cells, the per-half-latch cross-section is approximately the same as the per-configuration-bit cross-section. When only comparing the half latches that hold for one second or longer, the per-half-latch cross-section is one order of magnitude smaller than the per-configuration-bit cross-section.

Figure 5(a) shows the Virtex-4's distribution of hold times. The data remains bi-modal with peaks at 0.04 and 0.26 seconds. This figure shows the distribution of hold times for 15 MeV, 25 MeV, and 40 MeV beams at Texas A & M and there is no significant difference in hold times based on kinetic energy. On average, 99.5% of all half latches returned to their configured value within one second and 0.5% of all half latches did not return to their configured value after one second. The per-half latch cross-section is shown in Figure 5(b). This data shows that maximum per-half latch cross-section is $7.32E-009 \pm 9.08E-011$ $\frac{cm^2}{HL}$ for all half latches and is $1.35E-011 \pm 3.90E-012$ $\frac{cm^2}{HL}$ for half latches that persist for longer than one second, which shows a six times decrease in half latch cross-section from the Virtex-II to the Virtex-4. When this data is compared to the bit cross-section for configuration cells,

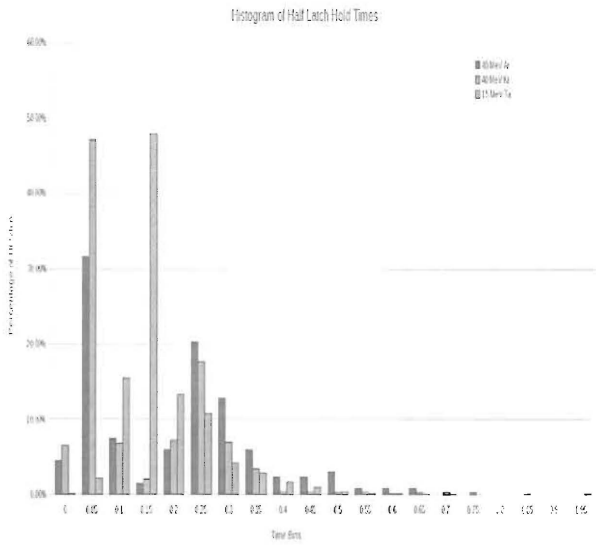


(a) Half-Latch Hold Times

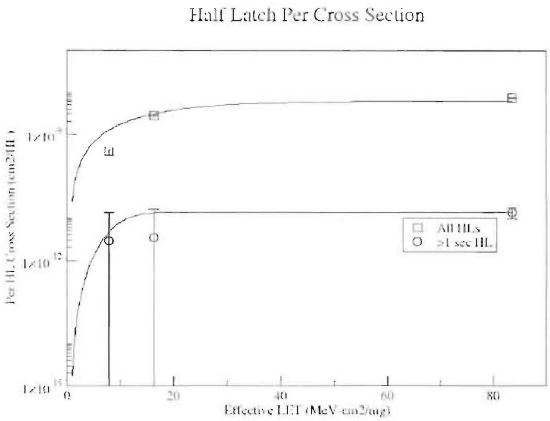


(b) Cross-section

Fig. 4. Virtex-II Half Latch Analysis



(a) Half-Latch Hold Times



(b) Cross-section

Fig. 5. Virtex-4 Half Latch Analysis

the per-half-latch cross-section is two to three orders of magnitude smaller than the per-configuration-bit cross-section. When only comparing the half latches that hold for one second or longer, the per-half-latch cross-section is five to seven orders of magnitude smaller than the per-configuration-bit cross-section.

B. Explicit Logical Constants

In the Virtex-I and Virtex-II devices, constant LUTs are used to tie off unused inputs for adders and multipliers, as well as providing design-specified constants. In the Virtex-4, using constant LUTs are not necessary, as the architectural post provides the same functionality to the design. Unfortunately, the design tools will still choose constant LUTs some times. Both the LUT and the LUT's routing are SEU-susceptible. Unlike half latches, constant LUTs can be repaired through on-line reconfiguration, and the amount of time the incorrect constant persists is based on the scrub rate of the system. Unfortunately, the Xilinx design flow tools load balance constant LUTs. While the load balancing does decrease the number of LUTs that are used in the design for logical constants, Figure 6 shows that load balancing causes constant LUTs to source multiple constants. Figure 6 shows that a constant is generated in a LUT in the far right configuration logic block (CLB), which is then used for two LUTs in the middle CLB and one LUT in the left CLB.

Designs that have been protected with triple-modular redundancy (TMR) have been shown to fail from single-bit upsets in both fault injection and beam testing when the logical constants are load balanced. The initial dis-

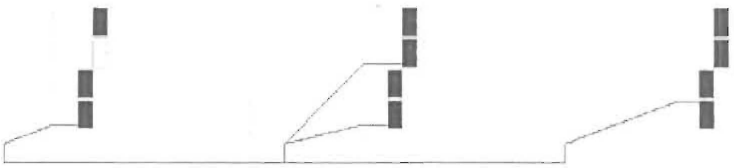


Fig. 6. Constant LUTs Schematics

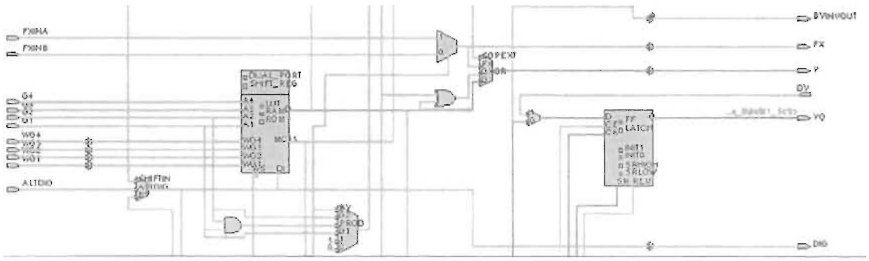


Fig. 7. Schematic of the G LUT with the GYMUX and Y Output in a CLB

covery of the constant LUTs was an adder tree circuit that was designed for the Virtex-II and used as part of the domain crossing error study [2]. In fault injection, 285 bits caused the design to output undetectable data errors when injected with single bit upsets. Some of these bits were later confirmed with accelerator testing. By translating the readback addresses of the single bit failures to physical locations, we were able to determine that all of the failures happened in CLBs where two or more TMR domains were present and a constant LUT was shared.

In further analysis, we were able to determine what was being disrupted in the circuit. We did this analysis with custom software that used JBits to compare two bitstreams. One bitstream is the original bitstream and the second bitstream has the corrupted bit changed. From this process we were able to identify two to four bits for each shared constant LUT that caused problems in the design. In Figure 7 the upper half of a Virtex-II CLB is shown. In this Figure, the GLUT, GYMUX and the Y output mux is show. While the LUT is flagged as being used, there are no inputs to the LUT, which is an indication that the LUT is being used as a constant. The output of the G LUT feeds through the GYMUX to be connected to the Y output mux. Two of the four bits that caused failures involve the GYMUX. In those cases, the GYMUX selects the FX and the F5 signals instead of the G LUT output signal. As the select lines for the GYMUX are sourcing configuration memory that has been programmed to determine which input to take, changing the select line values through SEUs is possible. The third bit was unidentifiable through this method, as

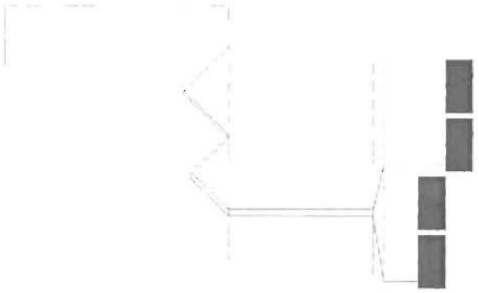


Fig. 8. Architectural Post Schematics

well as Xilinx's internal tools. We believe that it is either changing the LUT mode from LUT to shift register or is turning the YUSED buffer off. For these three bits, the errors are in predictable and regular locations throughout the device.

The forth bit that causes problems only happened twice in our reference design. The fourth bit has been identified as one of the output muxes available to the Y output signal in the CLB. In the two occasions where the bitflip caused a failure in a TMR-protected circuit, the change in the output mux caused the mux to select a Y output from another slice in the same CLB. Because the Y output changed value every clock cycle, it caused the circuit to fail. We believe these two failures are single-bit failure discovered several years ago [3].

In the Virtex-4, architectural posts are also used as a second type of explicit logical constants. A schematic of

how these posts are used is shown in Figure 8.[1] Despite the fact that Figure 8 shows that the post is driving two independent constants in the design, no single points of failure have been identified with this type of logical constants. In both accelerated testing and fault injection, no failures in TMR-protected designs can be tied to the architectural posts, despite having designs where two or more TMR domains share the same post. The only reliability concern with the Virtex-4 stems from the design tools choosing to use both the posts and the constant LUTs, including within the same CLB. In cases where the constant LUTs are being used, there are still instances where multiple domains share the same LUT.

III. Mitigation

There is no easy way to avoid logical constants at the design level. While it is possible to define all of the unused signals where a logical constant would be used, this approach would significantly change the design practices for many engineers. Furthermore, the load balancing of constant LUTs is done during the placement and routing part of the design flow tools and is not under the designer's control. Therefore, mitigating problems with both half latches and constant LUTs is best done after the design process. In this section, we present options and discussion regarding mitigating half latches and constants LUTs.

A. Recommendations on Mitigating Designs

The decision whether to mitigate half latches should be based on the device, the design, the mitigation, and the mission requirements. Our recommendation is to mitigate half latches for all Virtex-I designs, because these half latches do not reset. For the other devices, we recommend that only high-reliability applications should be mitigated. Furthermore, because half latches cannot drive multiple constants, upsets in the half latches of TMR-protected designs are less likely to be noticeable than upsets in half latches of unmitigated designs, as persistent corruption of state should not occur. Mitigating unprotected designs is not recommended as mitigation will increase the circuit's cross-section.

Constant LUTs, on the other hand, are only a problem in TMR-protected designs. Furthermore, because the Virtex-I lacks embedded multipliers or DSP cores, constant LUTs are more of a problem for the Virtex-II and Virtex-4. Therefore, it is our recommendation to mitigate TMR-protected Virtex-II and Virtex-4 designs and high-reliability, TMR-protected Virtex-I designs. Mitigating constant LUTs in unmitigated designs will increase the size of the design, thereby making the cross-section larger.

[1]While this is a similar schematic to the Virtex-II, the posts that were shown in earlier devices were abstractions of the half latches. In the Virtex-4, the posts are an abstraction of actual ground and VCC signals.

B. Mitigation Tools

There are three available tools for mitigating half latches. Two of the easiest ways to way to mitigate half latches is to use one of the two automated tools for applying TMR to user circuits (Xilinx's TMRTool [4] and Brigham Young University's BL-TMR Tool [5]). With both of these tools, the user can choose to extract half latches, which will cause *all* logical constants to be sourced from constant input pins. When using triplicated pins this method has been shown to mitigate both types of logical constants effectively.

The other option is to use the RADDRC tool from Los Alamos National Laboratory after the user circuit has been placed and routed. This tool has been modified recently to remove the load balancing on constant LUTs. The RADDRC tool works with the Xilinx design language (XDL) circuit representation to extract half latches to constant LUTs and then duplicates any shared constant LUTs to undo the load balancing. The mitigated XDL circuit will then need to be placed and routed a second time to route the new constant LUTs, but it is possible to preserve earlier timing and placement so that timing will not be adversely affected. This method has been shown in radiation testing to remove half latches. Fault injection testing has shown that single bit failures in the Virtex-II can be mitigated using this method, including a complete reduction of the single bit cross-section from the adder tree circuit from the domain crossing error study. A new version of the RADDRC tool will be available in December 2009 to ensure that all of the logical constants in Virtex-4 designs are extracted to the post.

IV. Conclusions

In conclusion, we have presented a number of reliability concerns with logical constants in the Xilinx Virtex family. There are two main categories of logical constants: implicit and explicit logical constants. In all of the Virtex devices, the implicit logical constants are implemented using half latches, which in the most recent devices are several orders of magnitudes smaller than configuration bit cells. Explicit logical constants are implemented exclusively using constant LUTs in the Virtex-I and Virtex-II, and use a combination of constant LUTs and architectural posts to the ground plane in the Virtex-4. We have also presented mitigation

methods and options for these devices. While SEUs in implicit and some types of explicit logical constants can cause data corrupt, the chance of failure from these components is now much smaller than it was in the Virtex-I device. Therefore, for many cases, mitigation might not be necessary, except under extremely high reliability situations.

V. Acknowledgments

The authors would like to acknowledge the input and support of Rocky Koga and Jeff George from The Aerospace Corporation without which this research would not be possible.

References

[1] P. Graham, M. Caffrey, M. Wirthlin, D. E. Johnson, and N. Rollins, "SEU mitigation for half-latches in Xilinx Virtex FPGAs," *IEEE Transactions on Nuclear Science*, Vol. 50, No. 6, pp. 2139–2146, December 2003.

[2] H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, and K. Lundgreen, "Domain Crossing Errors: Limitations on Single Device Triple-Modular Redundancy Circuits in Xilinx FPGAs," *IEEE Transactions on Nuclear Science*, Vol. 54, No. 6, pp. 2037 – 43, 2007.

[3] L. Sterpone and M. Violante, "A new reliability-oriented place and route algorithm for sram-based fpgas," *Transactions on Computers*, Vol. 55, No. 6, pp. 732 – 744, 2006.

[4] "Xilinx TMRTool User Guide," on web: http://www.xilinx.com/products/milaero/ug156.pdf.

[5] K. Morgan, M. Caffrey, P. Graham, E. Johnson, B. Pratt, and M. Wirthlin, "SEU-Induced Persistent Error Propagation in FPGAs," *IEEE Transactions on Nuclear Science*, Vol. 52, No. 6, pp. 2438 – 45, 2005.