LA-UR-

Approved for public release;
distribution is unlimited.

Title:	
Author(s):	
Intended for:	

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, 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. This abstract presents experimental data and mitigation methods for logical constants.

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.

Corresponding and Presenting Author:
Heather Quinn, Los Alamos National Laboratory, ISR-3 Space Data Systems, Los Alamos, NM, 87545 USA, phone: 505-665-7041, fax: 505-667-4197, e-mail: hquinn@lanl.gov

Contributing Authors:
Paul Graham and Keith Morgan, Los Alamos National Laboratory, ISR-3 Space Data Systems, Los Alamos, NM, 87545 USA, e-mail: grahamp@lanl.gov, morgank@lanl.gov

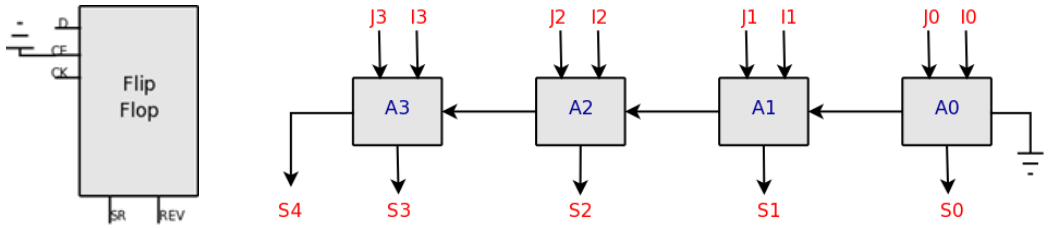
Patrick Ostler, Brigham Young University, Provo, UT, 84602 USA, email: ospatch@gmail.com

Greg Allen, Jet Propulsion Laboratory, 4800 Oak Grove Dr, Pasadena, CA USA, email: gregory.r.allen@jpl.nasa.gov

Gary Swift and Chen Wei Tseng, The Xilinx Corporation, 2100 Logic Drive, San Jose, CA, USA, email: gary.swift@xilinx.com, chenweit@xilinx.com

Session Preference: Single-event effects: devices and integrated circuits

Presentation Preference: Poster in either main conference or radiation data workshop



(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

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 abstract, 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. 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 are implemented with architectural posts that provide access to ground rails. As most of the logical constants 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, their failure can lead to the corruption of intermediate processing values and output data.

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. Reliability Concerns and Test Data

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.

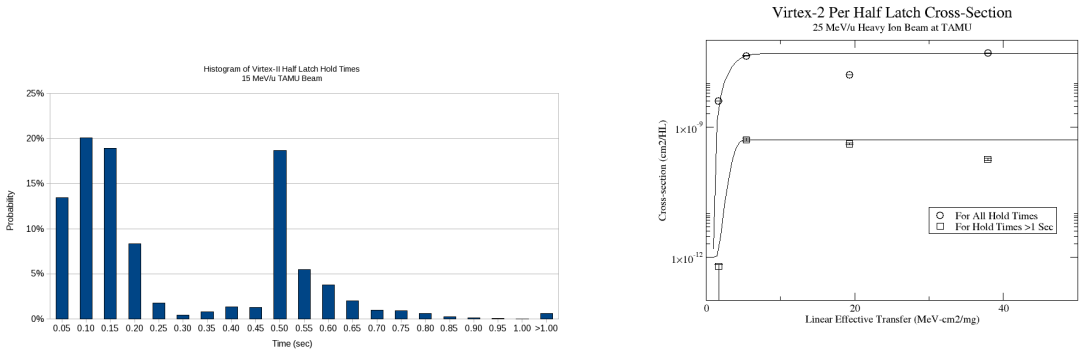
A. Implicit Logical Constants

Half latches were previously discussed in [1]. That paper presented data that showed that the Virtex-I half latches could not "scrubbed" through on-line reconfiguration, and only returned to the configured value through full off-line reconfiguration of the device. Because off-line reconfiguration interrupts the circuit's operation and observing thousands of half latches in a design was difficult, 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. Therefore testing was needed to determine both the half latch cross-section and how long the upset value would persist, called the *hold time*.

The Virtex-II, and Virtex-4 devices were tested at Texas A&M and Lawrence Berkeley National Laboratory cyclotrons. 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. Each device was tested with the same basic design based on 32 separate shift registers. The shift registers are built from user flip flops and the length of the

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



(a) Half-Latch Hold Times

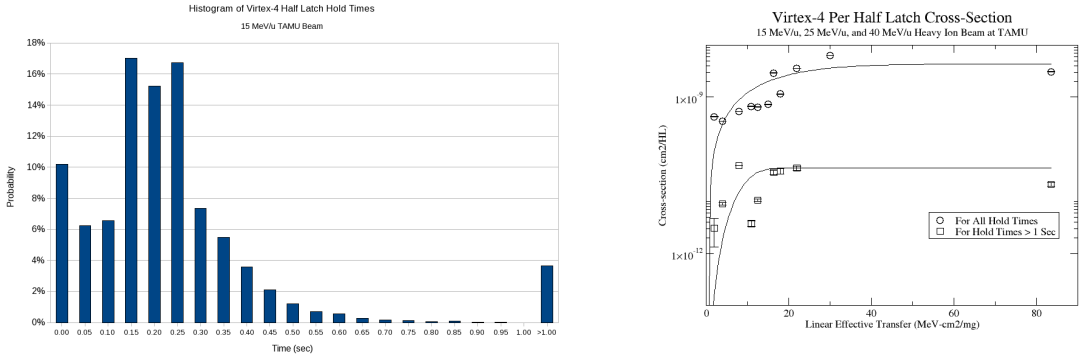
(b) Cross-section

Fig. 2. Virtex-II Half Latch Analysis

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 input strings of all ones and all zeros are alternated, which are inverted in the shift registers. Errors in the design can be detected through changes in the outputs. Because errors can be caused by direct strikes to the user flip flops, the user flip flop's half latches, or configuration memory, it is necessary to detect the difference between these types of faults. Therefore, faults that persist for less than two scrub cycles are considered not half latches, as errors in user flip flops or configuration memory should be fixed within one scrub cycle and the design resynchronized within another scrub cycle. Half latches that returned to their configured value within two scrub cycles will not be recorded. When a half latch is detected in a shift register chain, a counter will increment each clock cycle until the half latch clears. These counters are attached to the output and are sampled on average every 0.125 to 0.5 seconds to determine the current value of the counters. After a half latch returns to its configured value, the counter's final value persists for a few samples to ensure logging the value before being reset.

As shown in Figure 2(a), in the Virtex-II the distribution of hold times are bi-modal[1] 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 2(b). This data shows that 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. Figure 3(a) shows the Virtex-4's distribution of hold times. The data remains multi-modal with peaks at 0.04, 0.16, 0.18, 0.19, and 0.26 seconds. While this figure shows the distribution of hold times for 15 MeV/u, there is no significant change in hold times based on kinetic energy. On average, 96.4% of all half latches returned to their configured value within one second and 3.6% of all half latches did not return to their configured value after one second. The per-half latch cross-section is shown in Figure 3(b). This data shows that maximum per-half latch cross-section is $1.14E-009 \pm 2.61E-011$ $\frac{cm^2}{HL}$ for all half latches and is $1.05E-0116.37E-013 \pm 6.37E-013$ $\frac{cm^2}{HL}$ for half latches that persist for longer than one second, which shows a 44 times decrease in half latch cross-section from the Virtex-II to the Virtex-4.

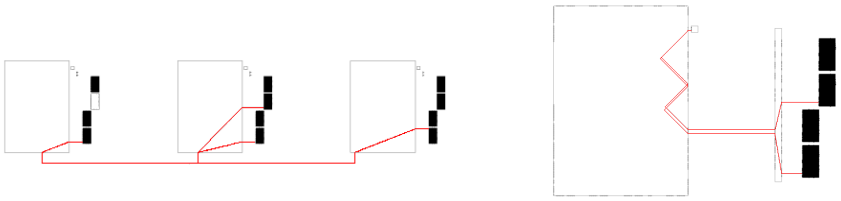
[1]We are still investigating why the data is multi-modal. Our current hypotheses are differences in half latch layout or differences in the counter designs. We are working with Xilinx and doing more in-depth analysis to determine if there is a logical reason for the two modes.



(a) Half-Latch Hold Times

(b) Cross-section

Fig. 3. Virtex-4 Half Latch Analysis



(a) Constant LUTs Schematic

(b) Architectural Posts Schematics

Fig. 4. Implementations of Explicit Logical Constants

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. 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 4(a) shows that load balancing causes constant LUTs to source multiple constants. Figure 4(a) 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 discovery 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 the Virtex-4, instead of using constant LUTs for explicit logical constants, architectural posts are used. A schematic of how these posts are used is shown in Figure 4(b).[2] Despite the fact that Figure 4(b) 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.

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

[2]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.

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 and the design being used. 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. Therefore, it is our recommendation to mitigate TMR-protected Virtex-II 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 design less reliable.

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 [3] and Brigham Young University's BL-TMR Tool [4]). In 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 LANL's RADDRC tool 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.

IV. Conclusions

In this abstract, we have presented a number of reliability concerns with logical constants in the Xilinx Virtex family. We have presented data for the cross-section and the hold times for half latches in the Virtex-II, and Virtex-4 devices. We have also presented mitigation methods and options for these devices.

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

[4] 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.