IROC Technologies
WTC POX 1510
Grenoble
Tel : +33.4.38.12.07.63
Fax : +33.4.38.12.96.15

SEE Test Report

Single Event Transient (SET) Measurement

MicroSemi A3P3000L FPGAs

Authors:

Adrian Evans, Dan Alexandrescu (IROC Technologies)

The work presented in this report was done under ESA Contract 4000107761/13/NL/CP

Technical Officer : Véronique Ferlet-Cavrois, ESA-ESTEC, TEC-QEC

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Acronym	Definition
DC	Don’t Care
DMR	Dual Modular Redundancy
DUT	Device Under Test
ECC	Error Correcting Code
HDL	Hardware Description Language
LET	Linear Energy Transfer
LFSR	Linear Feedback Shift Register
PID	Proportional, Integral, Derivative (algorithm for control systems)
SEC	Single Error Correct
SET	Single Event Transient
TMR	Triple Modular Redundancy

Revision History

Version	Date	Auhor	Comment
1	27-JUN-14	A. Evans	Initial draft sent for review.
2	1-JUL-14	A. Evans, D. Alexandrescu	Added event rate calculation for space mission.
3	13-NOV-14	A. Evans	Updates based on feedback from VFC received on 30-OCT and 31-OCT.
4	19-NOV-14	A. Evans	Removed watermark and confidential notice.

1 Introduction

Single Event Transients (SETs) are a serious concern in complex FPGAs and ASICs used in space applications. An accurate understanding of (i) their rate of occurrence, (ii) the distribution of pulse-widths and (iii) the deformation of the pulses as they propagate is essential in order to quantify and mitigate their effects. Knowledge of the maximum pulse width at the output of a network of combinatorial logic is required in order to correctly determine the temporal gap required to implement multiple sampling techniques (double sampling for detection or TMR for mitigation).

During this project, a series of specialized circuits were implemented on an A3P3000L Flash-Based FPGA and tested under heavy ion (HI) and pulsed laser. A detector, capable of performing accurate, on-chip SET pulse width measurements, was designed and validated. Using this sensor, the SET sensitivity of a number of different combinatorial Versatiles configurations was studied. The Versatile is the basic building block of MicroSemi ProAsic series FPGAs. A VeraTile can be configured to implement any three input logic function, a latch or a flip-flop. A schematic view of the Versatile is re-produced from RD2 in Figure 1.

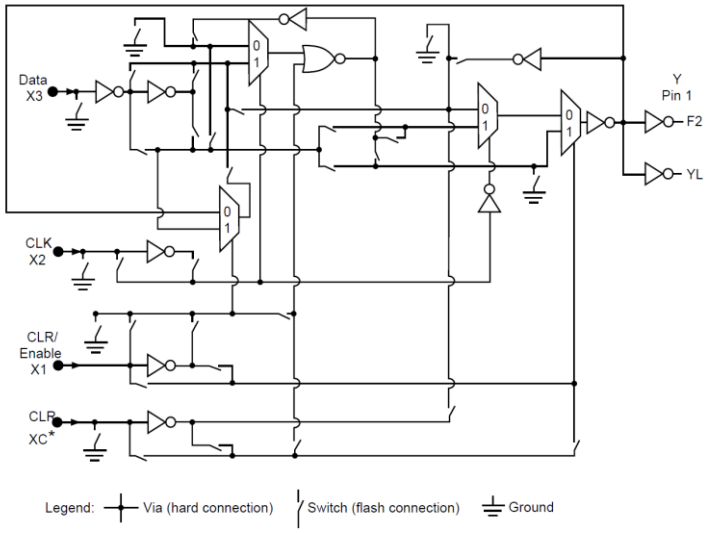

Figure 1 - Versatile Schematic

Using the SET detector, the SET sensitivity of different combinatorial networks was studied. These configurations were selected to investigate the effect of different Versatile configurations (BUF, AND2, MAJ3,..), the effect of the input states and the effect of pulse broadening and narrowing [RD10, RD20, RD21].

In addition to characterizing the SET sensitivity of simple Versatiles, a series of representative, complex logic circuits (e.g. adders, state-machine logic, etc.) were tested at speed in order to quantify the effect of SETs in realistic circuits. During these tests, the net effect of transients after logic de-ratings (LDR) and temporal de-rating (TDR) is observed. These cross sections are important when predicting the SET sensitivity of actual FPGA designs.

2 Reference Documents

	Document
RD1	R. Harada, Y. Mitsuyama, M. Hashimoto, and T. Onoye, "Measurement circuits for acquiring set pulsewidth distribution with sub-fo1-inverterdelay resolution," in Quality Electronic Design (ISQED), 2010 11th International Symposium on, March 2010, pp. 839–844.
RD2	MicroSemi Corporation, "Proasic3l fabric user guide," 2011.
RD3	C. Poivey, M. Grandjean, and F. X. Guerre, "Radiation characterization of microsemi proasic3 flash fpga family," in Radiation Effects Data Workshop (REDW), 2011 IEEE, 2011, pp. 1–5.
RD4	S. Rezgui, J. WANG, Y. Sun, B. Cronquist, and J. McCollum, "New reprogrammable and non-volatile radiation tolerant fpga: Rta3p," in Aerospace Conference, 2008 IEEE, 2008, pp. 1–11.
RD5	S. Rezgui, J. WANG, E. Tung, B. Cronquist, and J. McCollum, "New methodologies for set characterization and mitigation in flash-based fpgas," Nuclear Science, IEEE Transactions on, vol. 54, no. 6, pp. 2512–2524, 2007.
RD6	S. Rezgui, J. WANG, Y. Sun, B. Cronquist, and J. McCollum, "Configuration and routing effects on the set propagation in flash-based fpgas," Nuclear Science, IEEE Transactions on, vol. 55, no. 6, pp. 3328–3335, 2008.
RD7	N. Battezzati, L. Sterpone, and V. Ferlet-Cavrois, "Analysis of set propagation in flash-based fpgas by means of electrical pulse injection," in Radiation and Its Effects on Components and Systems (RADECS), 2009 European Conference on, 2009, pp. 87–90.
RD8	M. Berg, J.-J. Wang, R. Ladbury, S. Buchner, H. Kim, J. Howard, K. LaBel, A. Phan, T. Irwin, and M. Friendlich, "An analysis of single event upset dependencies on high frequency and architectural implementations within actel rtax-s family field programmable gate arrays," Nuclear Science, IEEE Transactions on, vol. 53, no. 6, pp. 3569–3574, 2006.
RD9	M. Berg, K. LaBel, H. Kim, M. Friendlich, A. Phan, and C. Perez, "A comprehensive methodology for complex field programmable gate array single event effects test and evaluation," Nuclear Science, IEEE Transactions on, vol. 56, no. 2, pp. 366–374, 2009.
RD10	V. Cavrois, V. Pouget, D. McMorrow, J. Schwank, N. Fel, F. Essely, R. S. Flores, P. Paillet, M. Gaillardin, D. Kobayashi, J. S. Melinger, O. Duhamel, P. Dodd, and M. Shaneyfelt, "Investigation of the propagation induced pulse broadening (pipb) effect on single event transients in soi and bulk inverter chains," Nuclear Science, IEEE Transactions on, vol. 55, no. 6, pp. 2842–2853, 2008.
RD11	L. Sterpone, N. Battezzati, and V. Ferlet-Cavrois, "Analysis of set propagation in flash-based fpgas by means of electrical pulse injection," Nuclear Science, IEEE Transactions on, vol. 57, no. 4, pp. 1820–1826, Aug 2010.
RD12	L. Sterpone and N. Battezzati, "On the mitigation of set broadening effects in integrated circuits," in Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on, April 2010, pp. 36–39.
RD13	L. Sterpone, N. Battezzati, F. Kastensmidt, and R. Chipana, "An analytical model of the propagation induced pulse broadening (pipb) effects on single event transient in flash-based fpgas," Nuclear Science, IEEE Transactions on, vol. 58, no. 5, pp. 2333–2340, Oct 2011.
RD14	P. Dudek, S. Szczepanski, and J. Hatfield, "A high-resolution cmos timeto- digital converter utilizing a vernier delay line," Solid-State Circuits, IEEE Journal of, vol. 35, no. 2, pp. 240–247, 2000.
RD15	Micro Semi. Total Ionizing Dose Test Report. No. 11T-RT3PE3000L-CG484-QJA2G. February 2011.

RD16	L. Tambara, F. Kastensmidt, T. Balen and M. Lubaszeski, "Total Ionizing Does in a Mixed-Signal Flash-based FPGA," IV Workshop on the Radiation Effects on Electronic and Photonic Devices for Aerospace Applications, October 2012. Available at: http://www.ieav.cta.br/werice2012/Anais%20werice.pdf.
RD17	T. Miyahira, G. Swift, "Evaluation of Radiation Effects in Flash Memories". Report Prepared by Jet Propulsion Laboratory. Available at http://klabs.org/richcontent/MAPLDCon98/Papers/c4_miyahira.pdf
RD18	A. Pelligrino, "Summary Actel ProAsic/3E Irradiation Tests". Available at: https://indico.cern.ch/event/239292/contribution/7/material/slides/0.pptx
RD19	Peyrard, P.-F.; Beutier, T.; Serres, O.; Chatry, C.; Ecoffet, R.; Rolland, G.; Boscher, D.; Bourdarie, S.; Inguimbert, C.; Calvel, P.; Mangeret, R., "A toolkit for space environment," Radiation and Its Effects on Components and Systems, 2003. RADECS 2003. Proceedings of the 7th European Conference on , vol., no., pp.639,641, 15-19 Sept. 2003.
RD20	V. Ferlet-Cavrois, P. Paillet, D. McMorrow, N. Fel, J. Baggio, S. Girard, O. Duhamel, J. S. Melinger, M. Gaillardin, J. R. Schwank, P. E. Dodd, M. R. Shaneyfelt, and J. A. Felix, "New insights into single event transient propagation in chains of inverters—Evidence for propagation-induced pulse broadening," IEEE Trans. Nucl. Sci., vol. 54, no. 6, pp. 2338–2346, Dec. 2007.
RD21	L.W. Massengill and P.W. Tuinenga, "Single-Event Transient Pulse Propagation in Digital CMOS," IEEE Trans. Nucl. Sci., vol. 55, no. 6, pp. 2861-2871, Dec. 2008.
RD22	Radecs 2009 Short Course. G. Berger. "Experimental Tools to Simulate Radiation Environments / Radiation Effects."

2.1 Test Facility

The radiation testing was performed at the *Heavy Ion Irradiation* Facility (HIF) at the *Université Catholique de Louvain (UCL)*. in Louvain-la-Neuve, Belgium [RD22]. The following table shows the ions that were used, their energy, range and LET.

Table 1 – Ions, Energies, Ranges and LET for HIF Low Penetration Cocktail

Ion	**Energy (MeV)**	**Range (um Si)**	**LET (Si) MeV (mg/cm^2)**
$^{15}N^{3+}$	62	3.11	2.12
$^{20}Ne^{4+}$	78	6.21	4.22
$^{40}Ar8^{+}$	150	15.31	10.39
$^{84}Kr^{17+}$	316	40	27.16
$^{132}Xe^{26+}$	420	36	69

3 Test Circuits and Test Setup

A common test-setup and interface infra-structure was used for all the device configurations. In all cases, the test setup was as shown in Figure 2. The signals for controlling and monitoring the test card are driven by IROC's tester card and driven along a flexible micro-coax cable to the card hosting the device under test (DUT). The tester and the power supplies for the test card are controlled by a test laptop which controls all the tests and records all the error events. A separate laptop is used to re-program the FPGA using MicroSemi's standard FPGA programmer. For the HI testing, the IROC tester, the FPGA programmer and the test-card were all inside the vacuum chamber.

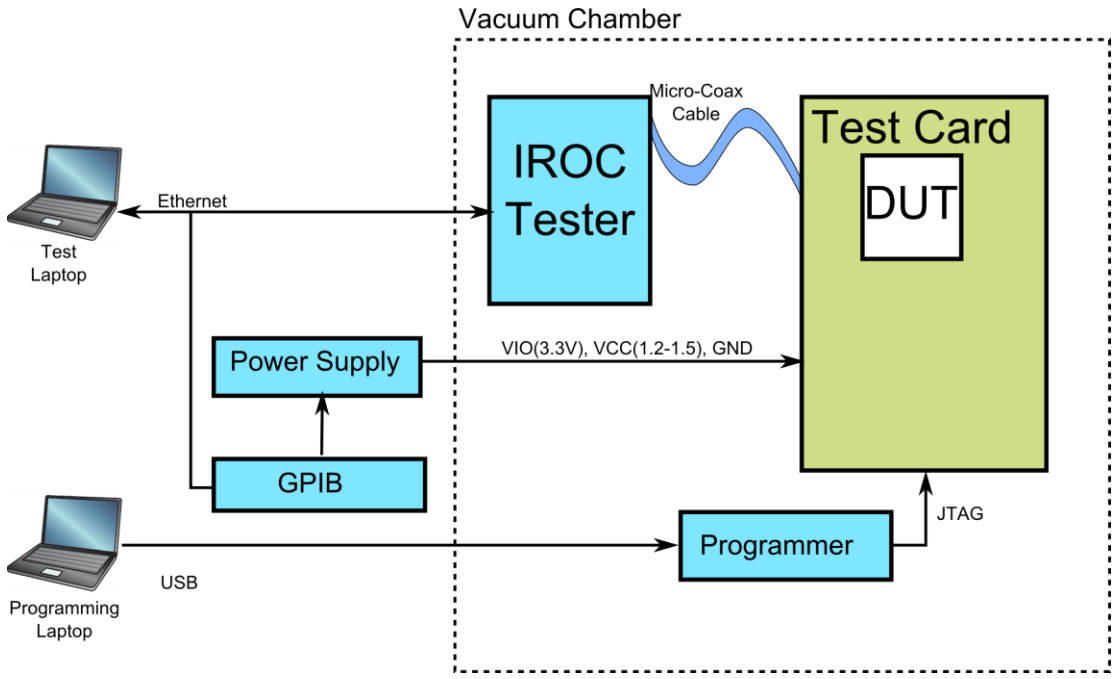

Figure 2 – Test Setup

3.1 Common Test Infra-Structure

In order to facilitate the design of multiple DUT FPGA configurations and to minimize the design effort, a common logical interface between the tester and the DUT was used. This interface is very general and can potentially be re-used for future radiation testing of FPGAs or dedicated test-chips.

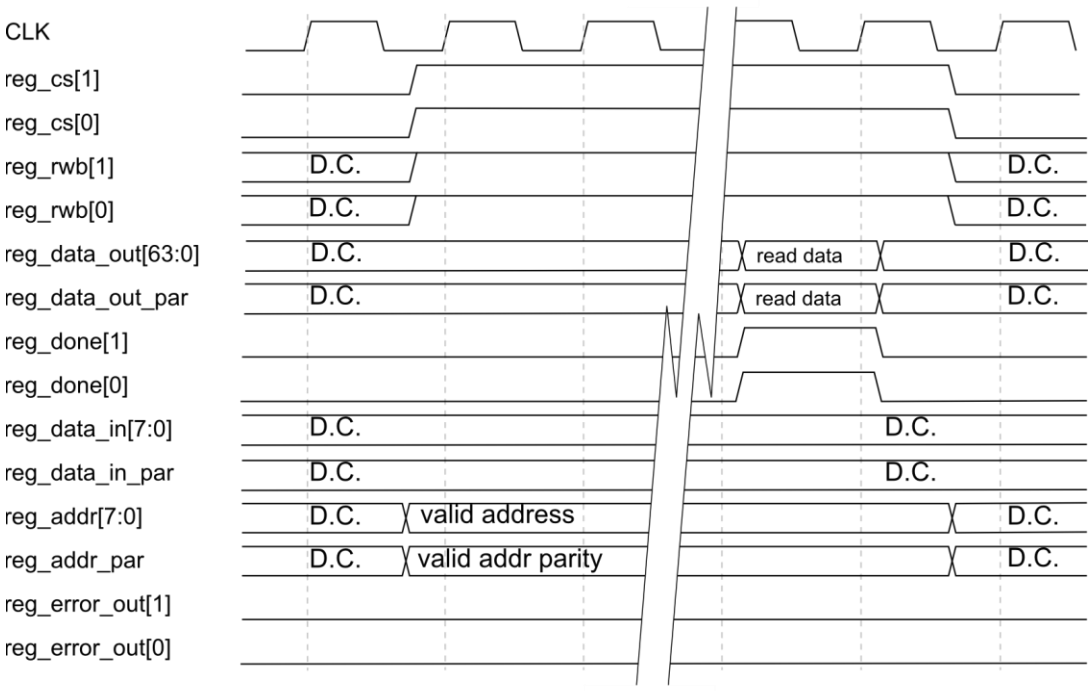

Figure 3 – Timing Diagram for Register Read Operation

With this interface, information is stored in or retrieved from the DUT using *register read* or *register write* operations. The actual status and configuration data stored in the DUT is held in TMR flip-flops and the register interface implements several protection techniques to ensure robust operation during radiation testing. In Figure 3and Figure 4 the timing diagrams for register read and register write operations are shown, respectively. Some key points about the interface are that the control signals (*reg_cs*, *reg_rwb*, *reg_done*) are duplicated and a transaction is only acted upon when both copies are consistent. We also notice that the read data, the write data and the address are protected by a parity bit. Transactions with bad parity are aborted. Finally, the data buses for read and write operations are asymmetric in width (64-bits for read, 8-bits for write). This choice was made since the amount of configuration (write) data is small, however, when errors occur, it is important to quickly extract the results (read data).

When the DUT detects a radiation-induced error, it flags this to the tester via an interrupt. The DUT can have multiple sensors and the interrupts are numbered such that the interrupt number corresponds to the number of the sensor that detected the error.

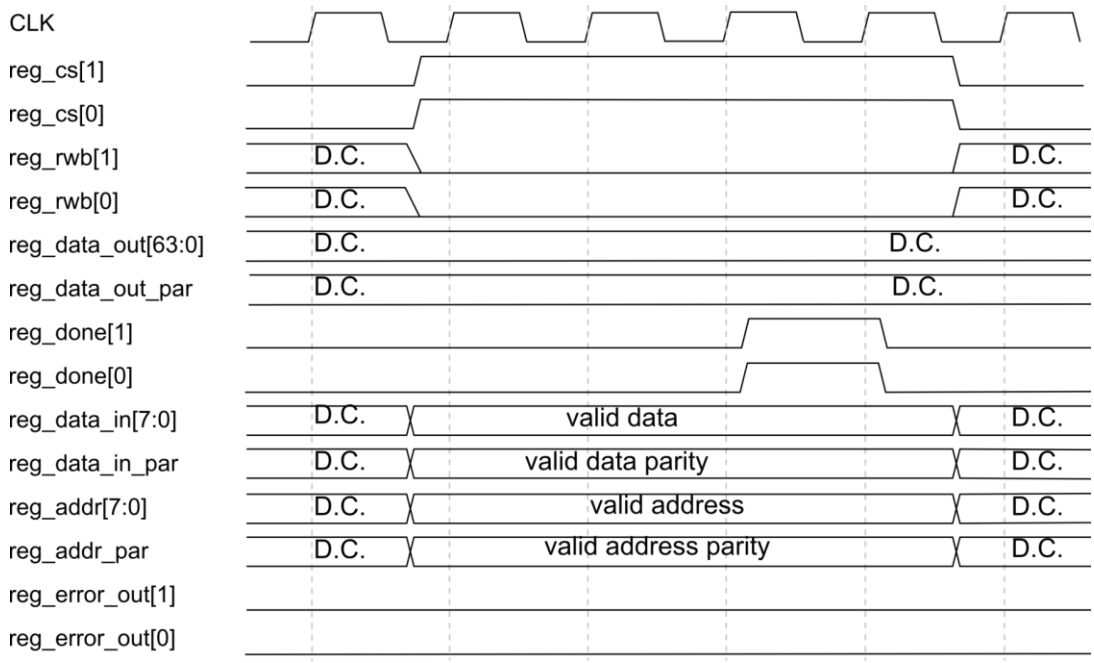

Figure 4 – Timing Diagram for Register Write Operation

When the DUT detects a radiation induced error, it flags the error to the tester by asserting an *interrupt*. Associated with the interrupt indication is an interrupt number. When the tester receives the interrupt indication, it knows which status register to read in order to extract the associated event information. The act of reading the interrupt status register clears the interrupt enabling the DUT to immediately respond to the next event.

Physically, 92 different detectors were placed in each DUT. The overall layout was as shown in Figure 5. The control logic (register and interrupt) was located in the centre of the die and the sensors were distributed as shown.

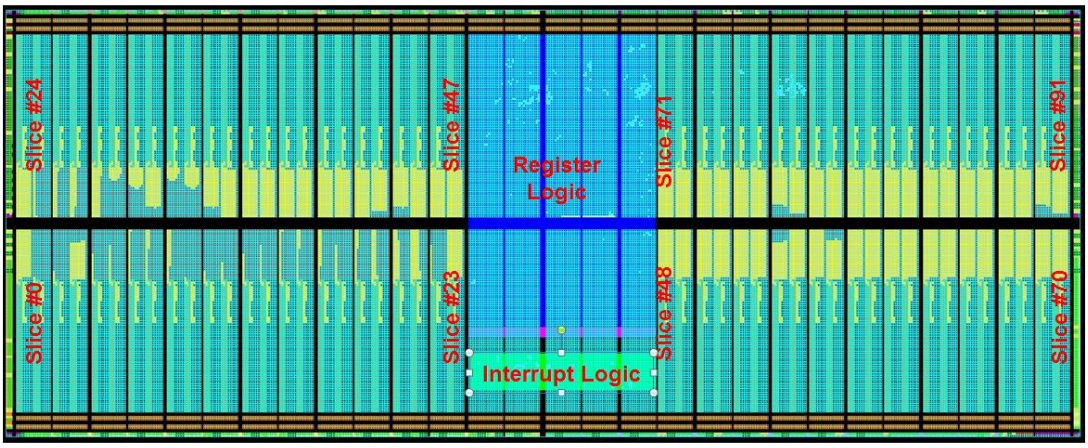

Figure 5 – Floorplan of Test FPGAs

3.2 Vernier SET Detector

The Vernier SET Detector circuit is based on exploiting the difference in delay between two gates to perform precise temporal measurements, a technique that was identified in RD14. The application of the Vernier technique measure the width of SETs was described in RD1 and is the basis of the circuit implemented in the course of this project.

The Vernier detector is shown in Figure 6 and the full HDL source code is reproduced in Appendix A. Upstream from the detector, is a combinatorial sensor circuit (see Section 3.4). When the incoming transient arrives at the detector, it drives two latches. The first one, *start latch*, has been preset to one in advance and the rising edge of the SET triggers the asynchronous CLEAR signal. The clearing of the *start latch* causes a falling edge to propagate along chain 1 which consists of a series of delay elements, each with a delay of t_1.

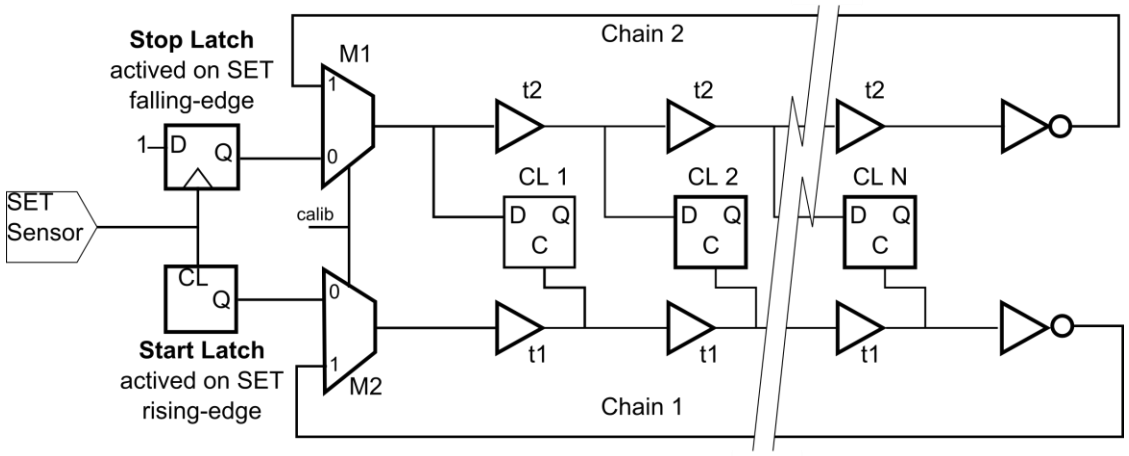

Figure 6 – Vernier Sensor

Later, when the falling edge of the SET arrives, it triggers the clock input of the *stop latch*. The D-input of the *stop latch* is tied to '1', thus the falling edge of the transient causes this value to be sampled and starts a rising edge propagating along chain 2 consisting of a series of delay elements, each with a delay of t_2.

The delay elements are selected such that $t_2 < t_1$, thus the rising edge on chain 2 catches up with the falling edge propagating on chain 1. At each stage, the edge flowing down chain 2 advances by a delay of (t2 - t1) compared to the edge on chain 1. The number of stages required for the two edges to meet depends on the width of the transient divided by (t2 - t1). The pulse width resolution depends on the difference in delay of two elements which can be made smaller than a single gate delay (≈500p..1000ps in ProAsic3L). In the current implementation the delay on the fast chain (2) was implemented using a BUFD and the delay on the slow chain (1) was implemented with an AO1 gate.

To detect the point where the two edges cross, a series of latches is used (CL1..CLN). Each stage on chain 1 drives the clock of a transparent latch. Initially, the *start latch* and all of chain 1 is high, thus all the capture latches are transparent. Because the output of the stop latch is initially low, as the falling edge on chain 1 propagates, the capture latches sample a '0' value. At the point when the rising edge of chain 2 catches up, the detection latches start to store the value '1'.

When either the *start* or *stop latch* change value, this triggers an interrupt to the external tester. The tester responds to the interrupt by reading the content of the latches (including the *start* and *stop* latches). By substituting input latches with opposite polarity inputs, the sensor can be used to measure negative ($0 \rightarrow 1$) pulses. In the HDL implementation a parameter controls whether the sensor is configured for positive or negative transients.

The minimum pulse width detection capabilities of this design are governed by the minimum valid pulse width for the *start* latch and the *stop* flip-flop. At 1.5V, the minimum pulse width for

an asynchronous clear (t_{WCLR}) is 340ps and the minimum pulse width for the flip-flop (t_{CKMPWH}) is 640ps.

3.2.1 Robustness of Vernier SET Detector

Since, the detector itself is on-chip, it is itself subject to upsets; however, the proposed design is actually very robust. Consider the following cases of upsets within the sensor:

- If a transient occurs in the delay elements on chain 1 it will cause the capture latches to momentarily become opaque but this does not change the stored value. No interrupt is triggered, thus the tester does not record an event.
- If a transient occurs in the delay elements on chain 2, the stored value in the capture latches will change momentarily but as they are transparent, this is restored by the output of the stop latch which is driving the chain.
- Note that after a SET correctly triggers the circuit, the *start* latch should hold a '0' value and the *stop* latch a '1' value. This is the signature of a correctly recorded SET event. If there is an SEU in either of the *start* or the *stop* latch, this will trigger an interrupt. In this case, the two latch values are inconsistent and the event is discarded from the log files. It is possible for a weak SET to trigger only one of the two latches and this situation is indistinguishable from an SEU in the *start* or *stop* latch. However, these false triggering cases represented <2.5% of all observed events under HI testing.

3.2.2 Delay Calibration

The actual delay of the VersaTiles depends on the voltage and to a lesser extent on the process variation. In order to ensure that the delays used for the SET measurement are calibrated, the SET delays in the SET sensor can be calibrated. This is done, by configuring the delay chains as Ring Oscillators (see M1 and M2 in figure Figure 6). The period of oscillation depends on the delay of the chain of gates (plus the additional delay from the inverter and the mux). Prior to radiation testing, the circuits are calibrated. Calibration results are shown in Section 4.1.

3.2.3 Placement of Vernier SET Detector

It is clear that controlling the timing and propagation delays is critical in order for the Vernier sensor to work correctly. For this reason, the sensor was designed as a hard macro (Design Block in the Libero Software package). Then, on the test-chip, multiple instances of the hard-macro were instantiated. This ensured that the cell placement was identical. The routing software attempts to preserve the routing of nets in the hard macro, but they may be modified during the global routing – which contributes to delay variation between the instances..

The placement of the cells of the Vernier SET sensor is shown in Figure 7. Chain 1 is on the left, the capture latches are in the middle and chain 2 is on the right. The start and stop latches are at the top.

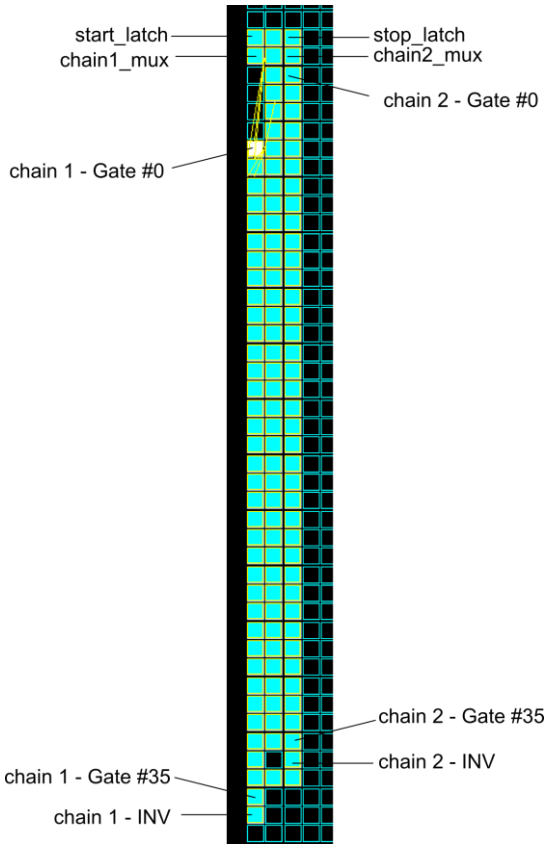

Figure 7 – Placement of Vernier SET Detector

3.3 Basic SET Detector

In order to validate the Vernier detector was correctly detecting transients, a second DUT was prepared using a Basic SET Detector which can only record the transient but not measure the pulse width. The Basic SET Detector consists of a single latch and is shown in Figure 8. The reset is connected to the clock input of the latch and when it is asserted causes the latch to store a one value. When a positive SET occurs, it causes the clear to be asserted and causes the latch output to return to zero.

This specific configuration was chosen because the minimum pulse width for an Asynchronous Clear (tWCLR) is nominally 340ps. For comparison, the minimum pulse width for a data clock (tCKMPWH) is nominally 640ps.

We note that the Basic SET Detector is NOT intrinsically robust. In other words, SEUs will alias as SETs. Most of the sensors that are connected to the detector contain a large number (50-100) combinatorial cells thus the combinatorial contribution dominants the SEU contribution.

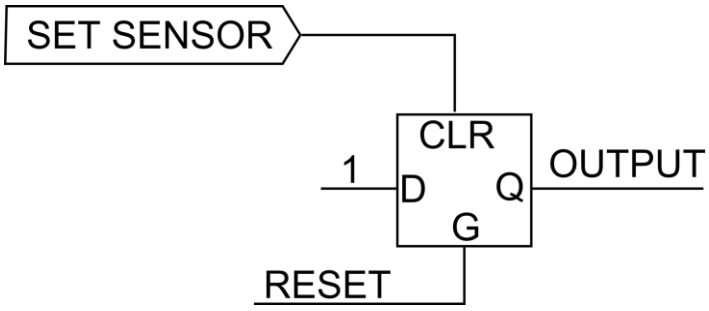

Figure 8 – Basic SET Detector

3.4 SET Sensor Configurations

It is known that the polarity of the SET and the type of interconnect between the VersaTiles can significantly affect the transient [RD6]. In order to gain a better understanding of these effects, many different SET sensor circuits were studied. These circuits can be divided into 3 categories based on their topology.

3.4.1 Linear Type 1 Sensor

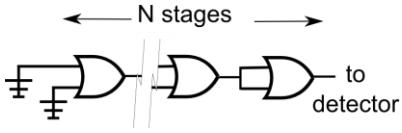

Figure 9 – Type 1 Linear Sensor

The simplest sensor is a chain sensor as shown in Figure 9. The output of one gate directly drives the inputs of the next gate. Obviously, the input value of the start of the chain, must be such that pulses propagate (for example a 00 input is required enable pulse propagation along a chain or OR gates).

3.4.2 Linear Type 2 Sensor

The problem with a simple chain sensor is that in order to have a sufficient number of gates, the chain must be extremely long. Long chains can excessively deform the original SET (broadening) and are not representative of real logic networks. For negative pulses, subject to narrowing, increasing the chain size has no benefit as pulses that occur at the start of the chain are suppressed due to narrowing.

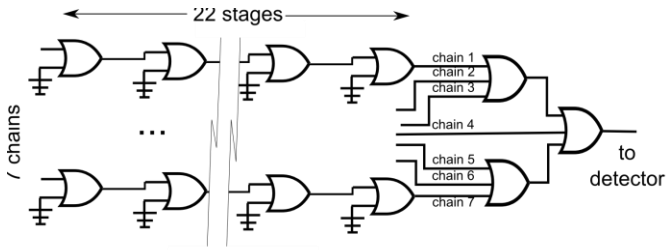

Figure 10 - Type 2 Linear Sensor

Thus, the Linear Type 2 Sensor, shown in, Figure 10 was also used. In this case, we chose to put seven parallel chains each of length 22 gates. The choice of seven chains was driven by physical constrains. The logic regions in the ProAsic3 are 16 cells wide so with seven parallel chains, two sensors can be placed in a region, while leaving a gap between adjacent sensors. The length of 22 gates was selected so that both the sensor and the Vernier detector could fit vertically in a logic region.

In order to converge the seven chains to a single output, two layers of OR3 gates are used. For sensors configured to detect negative SETs, two layers of AND3 are used instead. Unfortunately, these gates (OR3, AND3) are themselves sensitive to SETs and there is no exact means to precisely cancel their contribution. In total, this type of sensor has 7*22=154 gates in the chains and 3 gates to combine the chains. A simple correction factor of (154/157) was applied to the cross section calculation for sensors with this topology. This correction is not exact since the sensitivity of the OR3/AND3 may be different from the gates in the chains.

3.4.3 Tree Topologies

In order to address the shortcomings with the Type 2 Sensors, for gates whose inputs are symmetric, sensors with a tree topology were also used, as shown in Figure 11. With this type of structure, all the gates in the sensor are identical.

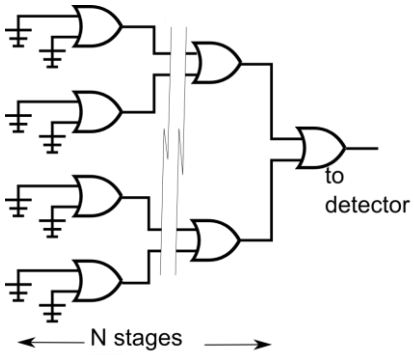

Figure 11 – Sensor with Tree Topology

3.4.4 Summary of Vernier Configurations

On the A3P3000L die there is room for 92 Vernier detectors and the set of sensors selected for the study is shown in the following table. The choice of complex gates was derived by profiling the Versatiles that were found to be used most frequently when synthesizing common circuits such as those described in section. 3.5. For each of the selected gate types, multiple configurations with different input patterns were selected in order to study the impact of gate inputs.

In addition to selecting a diverse set of complex gates, a series of simple gates (BUF) were configured in a set of chains. These chains were used to study the effect of pulse broadening. Finally, a set of detectors with identical BUF chains (sensors 0, 23, 24, 47, 48, 70, 71, 90) were instantiated as a consistency check.

Table 2 – Enumeration of Vernier SET Detectors

	Sensor Topology	***VersaTile***	***Input Values***	***Chained Input***	***SET***	***Comment***
Slice 0 : Die Edge – Large Linear Buffer Sensor						
0	7xLinear Type 1 7x22	BUF	0	Y	0->1->0	West Die Edge
Slice 1-8 : Study Pulse Broadening on Buffer Chain (0->1->0 SETs)						
1	Linear Type 1 Length 1	BUF	A=0	Y	0->1->0	
2	Linear Type 1 Length 4	BUF	A=0	Y	0->1->0	
3	Linear Type 1 Length 8	BUF	A=0	Y	0->1->0	
4	Linear Type 1 Length 12	BUF	A=0	Y	0->1->0	

	Sensor Topology	*VersaTile*	*Input Values*	*Chained Input*	*SET*	*Comment*
5	Linear Type 1 Length 16	BUF	A=0	Y	0->1->0	
6	Linear Type 1 Length 24	BUF	A=0	Y	0->1->0	
7	Linear Type 1 Length 32	BUF	A=0	Y	0->1->0	
8	Linear Type 1 Length 40	BUF	A=0	Y	0->1->0	
Slice 9-16 : Study Pulse Broadening on Buffer Chain (1->0->1 SETs)						
9	Linear Type 1 Length 1	BUF	A=0	Y	1->0->1	
10	Linear Type 1 Length 4	BUF	A=0	Y	1->0->1	
11	Linear Type 1 Length 8	BUF	A=0	Y	1->0->1	
12	Linear Type 1 Length 12	BUF	A=0	Y	1->0->1	
13	Linear Type 1 Length 16	BUF	A=0	Y	1->0->1	
14	Linear Type 1 Length 24	BUF	A=0	Y	1->0->1	
15	Linear Type 1 Length 32	BUF	A=0	Y	1->0->1	
16	Linear Type 1 Length 140	BUF	A=0	Y	1->0->1	
Slice 17-21 : Study Pulse Broadening on OR2 Versatile (0->1->0 SETs)						
17	Linear-Type 2 Length 4	OR2	B=0	A	0->1->0	
18	Linear-Type 2 Length 8	OR2	B=0	A	0->1->0	
19	Linear-Type 2 Length 12	OR2	B=0	A	0->1->0	
20	Linear-Type 2 Length 16	OR2	B=0	A	0->1->0	
21	Linear-Type 2 Length 24	OR2	B=0	A	0->1->0	
Slices 22 (+25-28) : Study Pulse Broadening on OR2 Versatile (1->0->1 SETs)						
22	Linear-Type 2 Length 4	OR2	B=0	A	1->0->1	

	Sensor Topology	*VersaTile*	*Input Values*	*Chained Input*	*SET*	*Comment*
Slice 23–24 : Die and Maintenance Region Edge – Large Linear Buffer Sensor						
23	7xLinear Type 1 7x22	BUF	0	Y	0->1->0	Border with Maintenance
24	7xLinear Type 1 7x22	BUF	0	Y	0->1->0	West Die Edge
Slices 25-28 (and 22) : Study Pulse Broadening on OR2 Versatile (1->0->1 SETs)						
25	Linear-Type 2 Length 8	OR2	B=0	A	1->0->1	
26	Linear-Type 2 Length 12	OR2	B=0	A	1->0->1	
27	Linear-Type 2 Length 16	OR2	B=0	A	1->0->1	
28	Linear-Type 2 Length 24	OR2	B=0	A	1->0->1	
Slices 29-31 – Study Binary Tree Sensor with OR2 Versatile (0->1->0 SETs)						
29	Binary Tree 31 gates	OR2	A=0 B=0	--	0->1->0	
30	Binary Tree 63 gates	OR2	A=0 B=0	--	0->1->0	
31	Binary Tree 127 gates	OR2	A=0 B=0	--	0->1->0	
Slices 32-34 – Study Binary Tree Sensor with AND2 Versatile (1->0->0 SETs)						
32	Binary Tree 31 gates	AND2	A=1 B=1	--	1->0->1	
33	Binary Tree 63 gates	AND2	A=1 B=1	--	1->0->1	
34	Binary Tree 127 gates	AND2	A=1 B=1	--	1->0->1	
Slices 35-40 – Study MX2 Versatile – Different Configurations						
35	7xLinear Type 2 7x22	MX2	A=0 B=1 S=0	A	0->1->0	Sensitive to S input
36	7xLinear Type 2 7x22	MX2	A=1 B=0 S=1	B	0->1->0	Sensitive to S input

	Sensor Topology	*VersaTile*	*Input Values*	*Chained Input*	*SET*	*Comment*
37	7xLinear Type 2 7x22	MX2	A=1 B=0 S=0	A	1->0->0	Sensitive to S input
38	7xLinear Type 2 7x22	MX2	A=0 B=1 S=1	B	1->0->1	Sensitive to S input
39	7xLinear Type 2 7x22	MX2	A=0 B=0 S=0	A	0->1->0	S input not sensitive
40	7xLinear Type 2 7x22	MX2	A=0 B=0 S=1	B	0->1->0	S input not sensitive
Slices 41-44 : Study XOR2 Versatile – Different Configurations						
41	7xLinear Type 2 7x22	XOR2	A=0 B=0	A	0->1->0	
42	7xLinear Type 2 7x22	XOR2	A=0 B=0	B	0->1->0	
43	7xLinear Type 2 7x19	XOR2	A=1 B=0	A	1->0->1	
44	Binary Tree 127 gates	XOR2	A=0 B=0	–	0->1->0	
Slices 45-46 : Study NOR2A Versatile– Different Configurations						
45	7xLinear Type 2 7x22	NOR2A	A=1 B=1	B	0->0->1 1->0->1	Alternating Pattern 010101...
46	7xLinear Type 2 7x22	NOR2A	A=0 B=0	A	0->1->0	
Slices 47-48 : Maintenance Region Edge – Large Linear Buffer Sensor						
47	7xLinear Type 1 7x22	BUF	0	Y	0->1->0	Border with Maintenance
48	7xLinear Type 1 7x22	BUF	0	Y	0->1->0	West Die Edge
Slices 49-54 : Study XOR3 Versatile – Different Configurations						
49	7xLinear Type 1	XOR3	A=0	--	0->1->0	

	Sensor Topology	*VersaTile*	*Input Values*	*Chained Input*	*SET*	*Comment*
	7x22		B=0 C=0			
50	7xLinear Type 1 7x22	XOR3	A=1 B=1 C=1	--	1->0->1	
51	7xLinear Type 2 7x22	XOR3	A=1 B=1 C=1	A	1->0->1	
52	7xLinear Type 2 7x22	XOR3	A=1 B=1 C=1	B	1->0->1	
53	7xLinear Type 2 7x22	XOR3	A=1 B=1 C=1	C	1->0->1	
54	Ternary Tree 121 gates	XOR3	A=0 B=0 C=0	–	0->1->0	
Slices 55-57 : Study NOR2B Versatile – Different Configurations						
55	7xLinear Type 1 7x22	NOR2B	A=0 B=0	–	0->1->0	
56	7xLinear Type 2 7x22	NOR2B	A=1 B=0	B	0->1->0	
57	Binary Tree 127 gates	NOR2B	A=1 B=1	--	1->0->1	
Slices 58-60 : Study AO1 – Different Configurations						
58	7xLinear Type 2 7x22	AO1	A=0 B=0 C=0	C	0->1->0	AND part not sensitive (A=0,B=0)
59	7xLinear Type 2 7x22	AO1	A=0 B=1 C=0	A	0->1->0	AND part is sensitive (B=1)
60	7xLinear Type 2 7x22	AO1	A=1 B=0 C=1	C	1->0->1	AND part is sensitive (A=0,B=0)
Slices 61-66 : Study MAJ3 Versatile – Different Configurations						
61	7xLinear Type 2 7x22	MAJ3	A=0 B=0 C=1	A	0->1->0	

	Sensor Topology	*VersaTile*	*Input Values*	*Chained Input*	*SET*	*Comment*
62	7xLinear Type 2 7x22	MAJ3	A=1 B=0 C=1	A	1->0->1	
63	7xLinear Type 2 7x22	MAJ3	A=0 B=0 C=1	B	0->1->0	
64	7xLinear Type 2 7x22	MAJ3	A=0 B=1 C=1	B	0->1->0	
65	7xLinear Type 2 7x22	MAJ3	A=0 B=1 C=0	C	0->1->0	
66	7xLinear Type 2 7x22	MAJ3	A=0 B=1 C=1	C	1->1->0	
Slices 67-69 : Study XA1 Versatile – Different Configurations						
67	7xLinear Type 2 7x22	XA1	A=0 B=0 C=1	A	0->1->0	XOR gate is sensitive
68	7xLinear Type 2 7x22	XA1	A=0 B=0 C=1	B	0->1->0	XOR gate is sensitive
69	7xLinear Type 2 7x22	XA1	A=1 B=0 C=1	C	1->0->1	XOR gate is sensitive
Slice 70–71 : Die and Maintenance Region Edge – Large Linear Buffer Sensor						
70	7xLinear Type 1 7x22	BUF	0	Y	0->1->0	East Die Edge
71	7xLinear Type 1 7x22	BUF	0	Y	0->1->0	Border with Maintenance
Slices 72-75 : Study OR3 Versatile – Different Configurations						
72	7xLinear Type 1 7x22	OR3	A=0 B=0 C=0	–	0->1->0	
73	7xLinear Type 2	OR3	A=1 B=0	A	1->0->1	

	Sensor Topology	***VersaTile***	***Input Values***	***Chained Input***	***SET***	***Comment***
	7x22		C=0			
74	7xLinear Type 2 7x22	OR3	A=0 B=0 C=0	B	0->1->0	
75	7xLinear Type 2 7x22	OR3	A=0 B=0 C=0	C	0->1->0	
76	Ternary Tree 121 gates	OR3	A=0 B=0 C=0	-		
Slices 77 : Study OR2A Versatile – Single Configuration						
77	7xLinear Type 2 7x22	OR2A	A=0 B=0	A	0->1->0 1->0->1	Alternating Pattern 010101...
Slices 78-83 : Study NOR3B Versatile – Different Configurations						
78	7xLinear Type 2 7x22	NOR3B	A=0 B=1 C=0	A	0->1->0	
79	7xLinear Type 2 7x22	NOR3B	A=1 B=1 C=0	A	1->0->1	
80	7xLinear Type 2 7x22	NOR3B	A=1 B=0 C=0	B	0->1->0	
81	7xLinear Type 2 7x22	NOR3B	A=1 B=1 C=0	B	1->0->1	
82	7xLinear Type 2 7x22	NOR3B	A=1 B=1 C=1	C	1->0->1	Alternating Pattern 010101...
Slices 83-86 : Study AX1B Versatile – Different Configurations						
83	7xLinear Type 2 7x22	AX1B	A=1 B=1 C=0	C	0->1->0	
84	7xLinear Type 2	AX1B	A=1 B=1	C	0->1->0	

	Sensor Topology	*VersaTile*	*Input Values*	*Chained Input*	*SET*	*Comment*
	7x22		C=0			
85	7xLinear Type 2 7x22	AX1B	A=1 B=1 C=1	C	1->0->1	AND gate not sensitive
86	7xLinear Type 2 7x22	AX1B	A=0 B=1 C=1	C	1->0->1	AND gate is senstitive
Slices 87-90 : Study AX1B – Different Configurations						
87	7xLinear Type 1 7x22	NOR3C	A=1 B=1 C=1	-	1->0->1	
88	7xLinear Type 2 7x22	NOR3C	A=1 B=1 C=1	A	1->0->1	
89	7xLinear Type 2 7x22	NOR3C	A=0 B=1 C=1	A	0->1->0	
90	Ternary Tree 121 gates	NOR3C				
Slice 90 : Die Region Edge						
91	7xLinear Type 1 7x22	BUF	0	Y	0->1->0	East Die Edge

3.5 Complex Circuits

In addition to the simple gate structures described in section 3.4, eight different complex circuits were tested under dynamic operation. To ensure realistic operating conditions, the pseudo-random input vectors were applied to the circuits generated using a Linear Feedback Shift Register (LFSR). Table 3 lists the circuits that were studied.

Table 3 – Summary of Complex Circuits

Number	**Name**	**Inputs**	**Outputs**	**Number Versatiles**
1	16-bit Carry Look Ahead Adder with CI	33	33	136
2	32-bit Priority Encoder	32	6	76
3	8-State State Machine	25	12	57

4	16-bit Magnitude Comparator	32	2	81
5	4-bit 8:1 Mux	52	3	28
6	16-bit Hamming ECC Encoder	16	5	20
7	16-bit Hamming ECC Decoder	21	16	60

In order to distinguish between the effect of SEUs and SETs, the complex circuits were tested as shown in Figure 12. Each circuit is instantiated three times. All three copies are driven with random vectors from the LFSR. For each output bit, there is a three way comparison between the copies and the result of the comparison is captured in flip-flops. If a SET affects one of the three circuits, then a two bit difference will be observed (e.g. a SET in CCT_0 will cause CCT_0 != CCT1 and CCT_0 != CCT2). If, however, there is a SEU in one of the flip-flops, only one bit will be upset. In this way, the effect of SETs can be differentiated from SEUs.

The N-bit LFSR cycles through the (N-1) valid states. Normally, a SEU would make the LFSR transition to another valid state, which is of no importance as it is generating pseudo-random vectors. There is a very small chance that an SEU would cause the LFSR to enter the all zeroes state which is a terminal state. During the radiation testing, this risk was mitigated by periodically resetting the DUT (once per second).

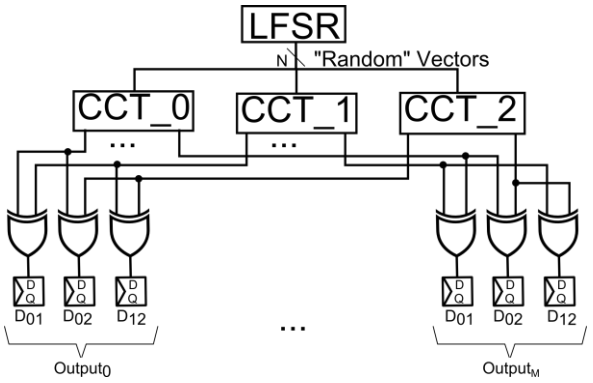

Figure 12 – Instantiation of Complex Circuits

3.5.1 Implementation of Complex Circuits

The HDL source code for each of the complex circuits is provided in Appendix B. Care was required to ensure a consistent implementation. First, the basic circuits (adder, multiplier, etc.) were synthesized, placed, routed and compiled into a hard-macro (Design Block). Figure 13 shows the hard macro for the 16-bit adder. The use of a hard-macro ensures that each of the three copies of the circuit has a nearly identical implementation.

Next, the full circuit shown in Figure 12 was generated and compiled into a hard macro (Design Block). Figure 14 shows the hard-macro created to implement three instances of the adder circuit. The logic for the LFSR is at the top (purple region) and the logic for performing the three way comparison of the output vectors is in the bottom (purple region).

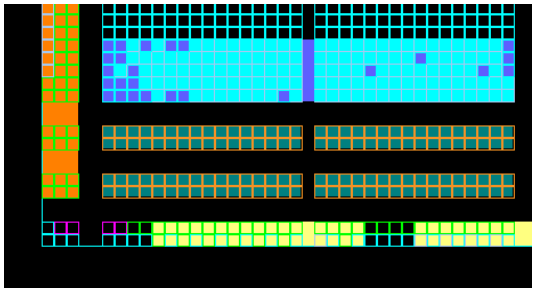

Figure 13 – Single 16-bit Adder

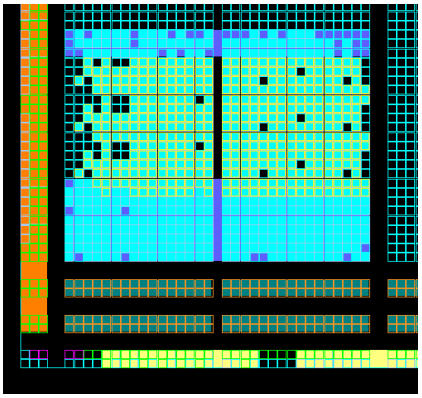

Figure 14- Three Instances of Adder

3.6 Device Preparation

For the Heavy-Ion testing, the devices were opened by our subcontractor, SERMA Technologies.

Devices used for heavy ions testing were opened on the front-side while laser testing required an opening on the back-side. Opening the device on both sides was not possible, due to issues related to the mechanical support of the die and bonding, even if the DUT board allows both types of tests to be performed on the same component.

The component preparation for heavy ions task had the objective of performing a manual front-side package opening until silicon by keeping the electrical integrity of the product and the PCB. The following operations were performed:

- Laser pre-decapsulation on frontside of the sample
- Manual Wet chemical finishing dedicated to standard wire.
- Coating removal
- Die cleaning using non-abrasive product

The following list of facilities and equipment have been used:

- Chemical facilities of SERMA Technologies laboratory.
- Laser SESAME 1000S
- Laser JetEtch SESAME 777Cu

Overall, the quality of this service was adequate for the application. All the prepared devices were operational after decapsulation. However, we observed some minor issues related to the cleanliness of the die at the end of the process (dust/residues). We have flagged this issue to SERMA to enable them to improve their process.

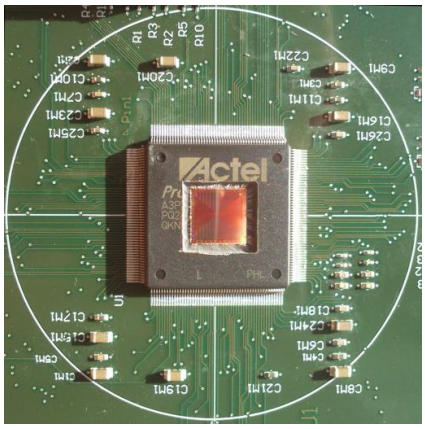

Figure 15 –A3PL3000L Device Opened on Top Side for HI Testing

4 Heavy Ion Test Results

Three FPGA configurations were tested at the Heavy Ion Facility at Leuven-La-Neuve (Belgium) on November 18th and 19th 2013. Four different FPGA configurations were tested, as listed in Table 4.

Table 4 – DUT Types

DUT	**Description**
DUT 1	92 x Vernier Sensors
DUT 3	92 x Basic Sensor
DUT 4	Complex Circuits – Static SET Measurement
DUT 6	Complex Circuits – Dynamic Operation

4.1 Calibration of Vernier Sensors

Prior to radiation testing and prior to any change in conditions (e.g. temperature, voltage or device), a calibration sequence was performed where the delay of each chain in the Vernier sensors was measured by configuring it as a ring-oscillator and measuring the frequency versus the reference clock. Each of the 92 sensors in DUT1 was calibrated individually. The table below shows the average delay across the 92 sensors at three voltage and three temperature conditions. The variation highlights the need for calibration when performing on-chip pulse delay measurements.

Each Vernier sensor had a chain with 36 delay elements. However, when the chain is configured as a ring-oscillator, there is the additional delay of the 2:1 mux at the start, the inverter at the end of the chain and the additional routing to close the loop (see Figure 6). In order to convert the average delay per-gate on the chain, the total oscillation period was divided by 39 (36 + 3) – in order to compensate for these additional delays. This uncontrolled, additional delay introduces some error in the delay measurement; however, it is relatively small (compared to the delay of the main part of the chain).

Table 5 – Delay Line Calibration

Temp. °C	1.08V			1.2V			1.5V		
	Chain 1 Gate Delay (ps)	Chain 2 Gate Delay (ps)	Delta Delay (ps)	Chain 1 Gate Delay (ps)	Chain 2 Gate Delay (ps)	Delta Delay (ps)	Chain 1 Gate Delay (ps)	Chain 2 Gate Delay (ps)	Delta Delay (ps)
25	890	736	154	772	628	144	575	464	111
40	912	752	160	783	639	144	588	474	114
70	946	780	166	825	674	151	615	497	118

At a given operating condition, there is also significant variation between the sensors. This uncertainty comes from on-die variation as well as from the fact that the global routing was not identical across the sensors. In Figure 16, the distribution of the delta gate delay (Vernier Resolution) for the 92 sensors is shown for three temperatures, at 1.2V operation.

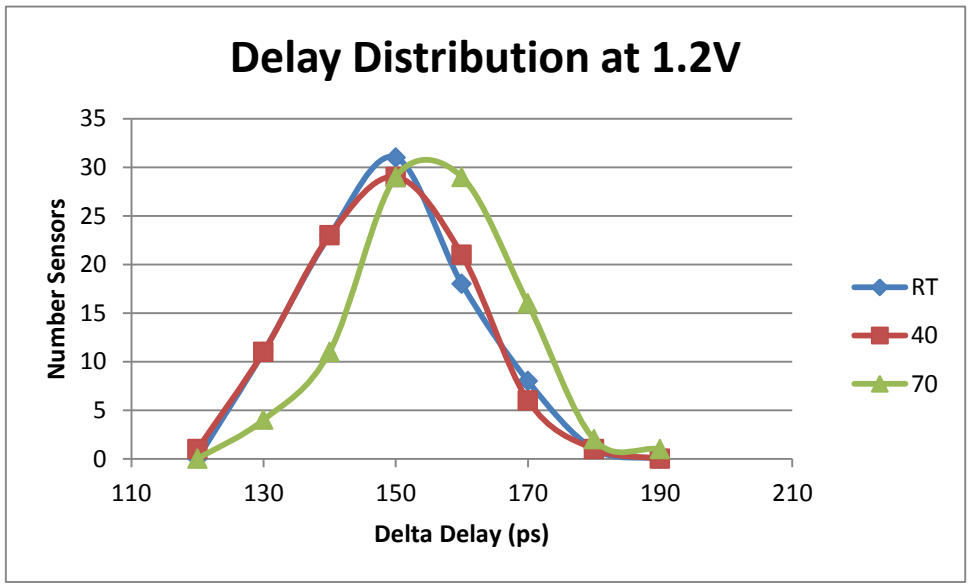

Figure 16 – Distribution of Vernier Sensor Measurement at 1.2V

4.2 Correlation between Vernier and Basic Sensor

In order to ensure that the Vernier sensor was indeed detecting transients and that the rate was correct, the cross sections measured by the Vernier Sensor were compared with those measured by the Basic detector. In Figure 17 through Figure 22, the measured cross section of several different Versatiles are shown with results from both the Vernier (DUT 1) and the Basic (DUT3) detectors. In all cases, the observed cross sections correspond, within the uncertainties (95% confidence error bars).

A better overall view of the correlation is shown in Figure 23 which is a scatter plot the measured cross section for a large number of detectors, with the measurement from the Vernier Detector on the x-axis and the measurement from the Basic Detector on the y-axis.

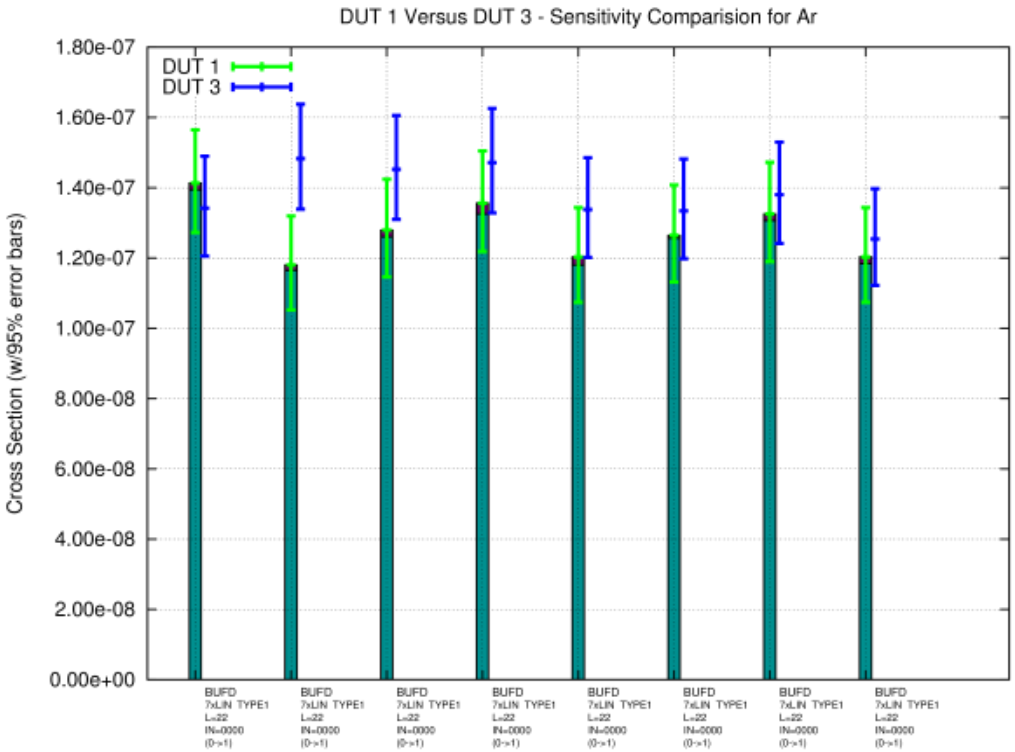

Figure 17 – DUT1,3 Correlation BUF (Ar)

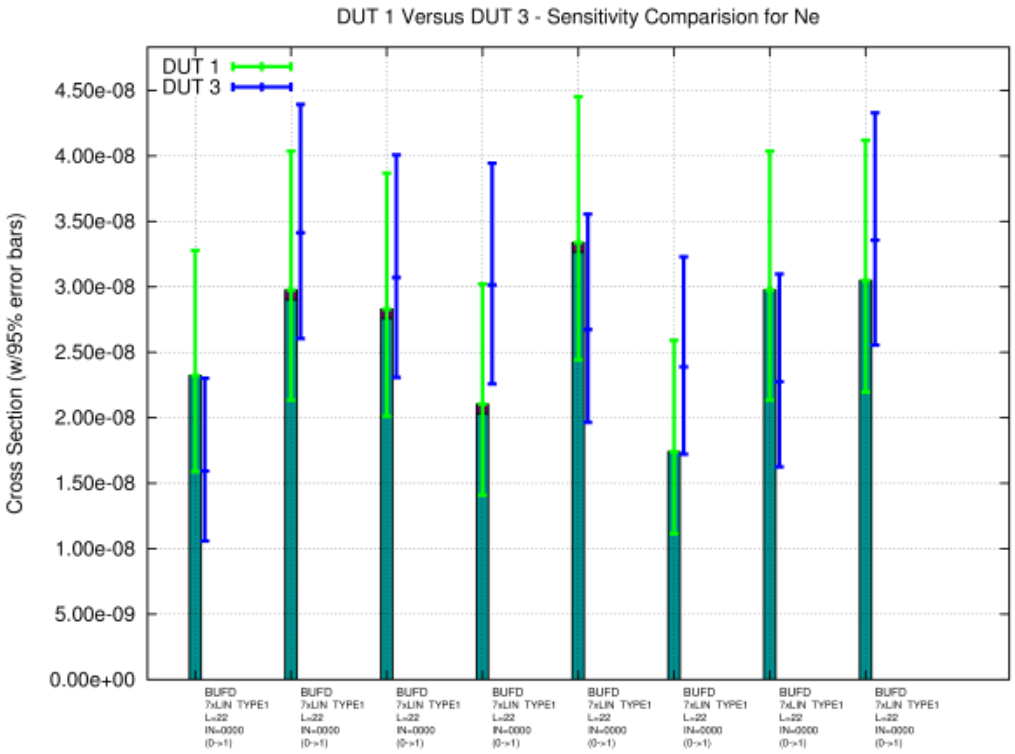

Figure 18 – DUT1,3 Correlation BUF (Ne)

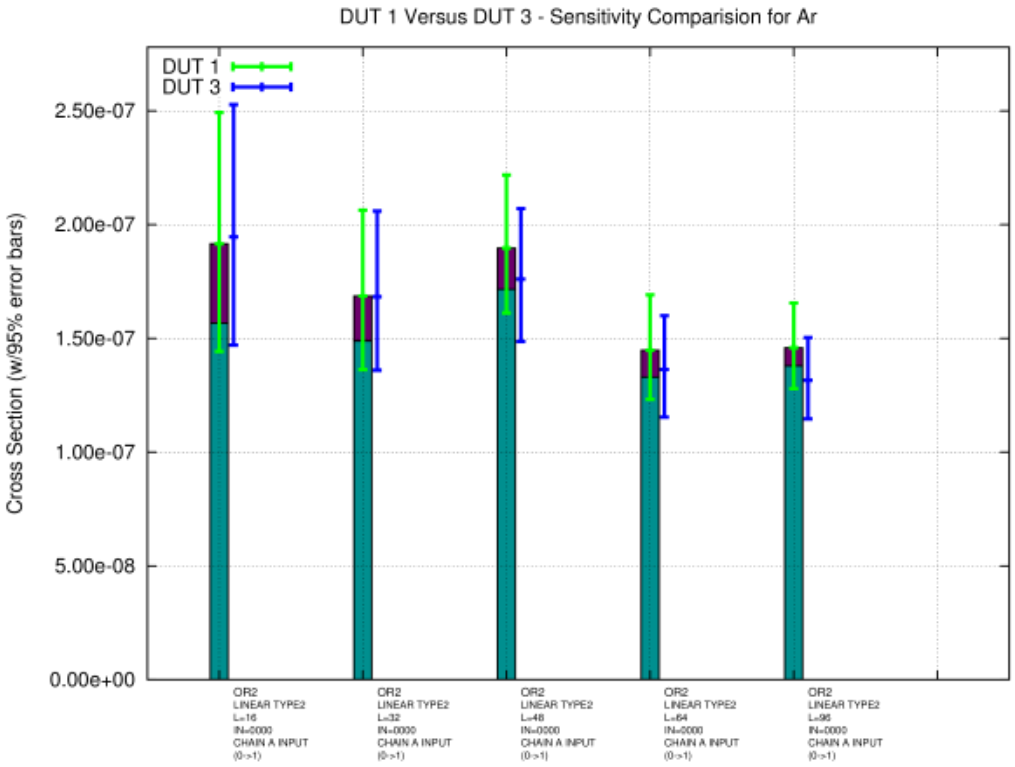

Figure 19 – DUT1,3 Correlation OR2(Ar)

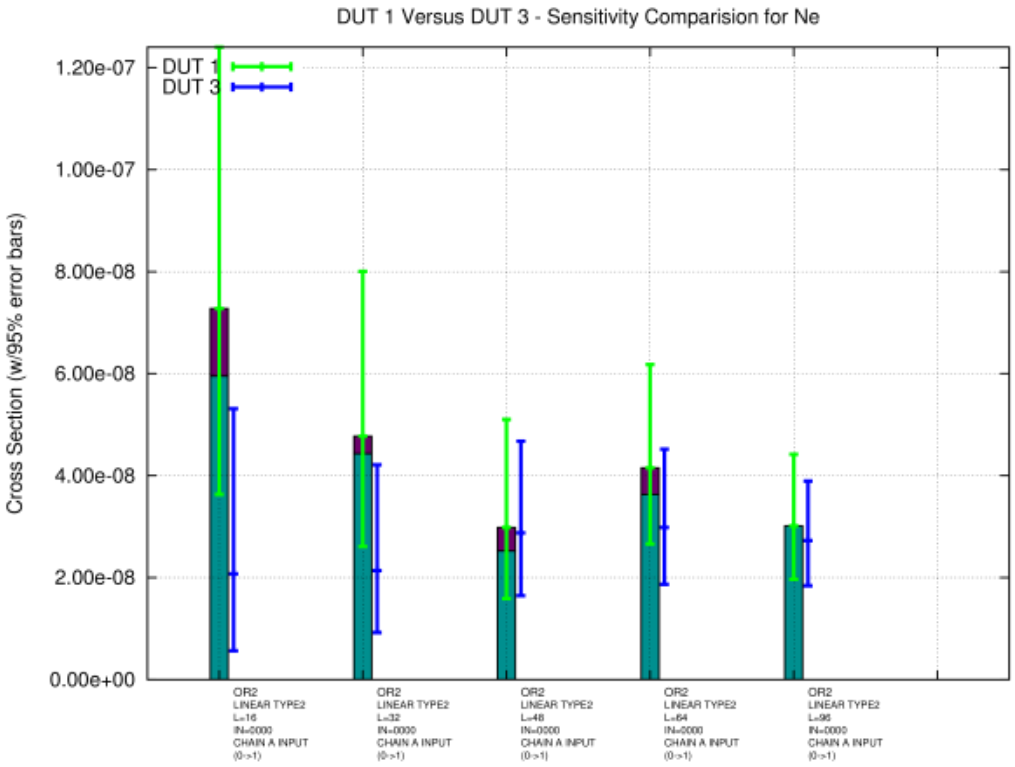

Figure 20 – DUT1,3 Correlation OR2(Ne)

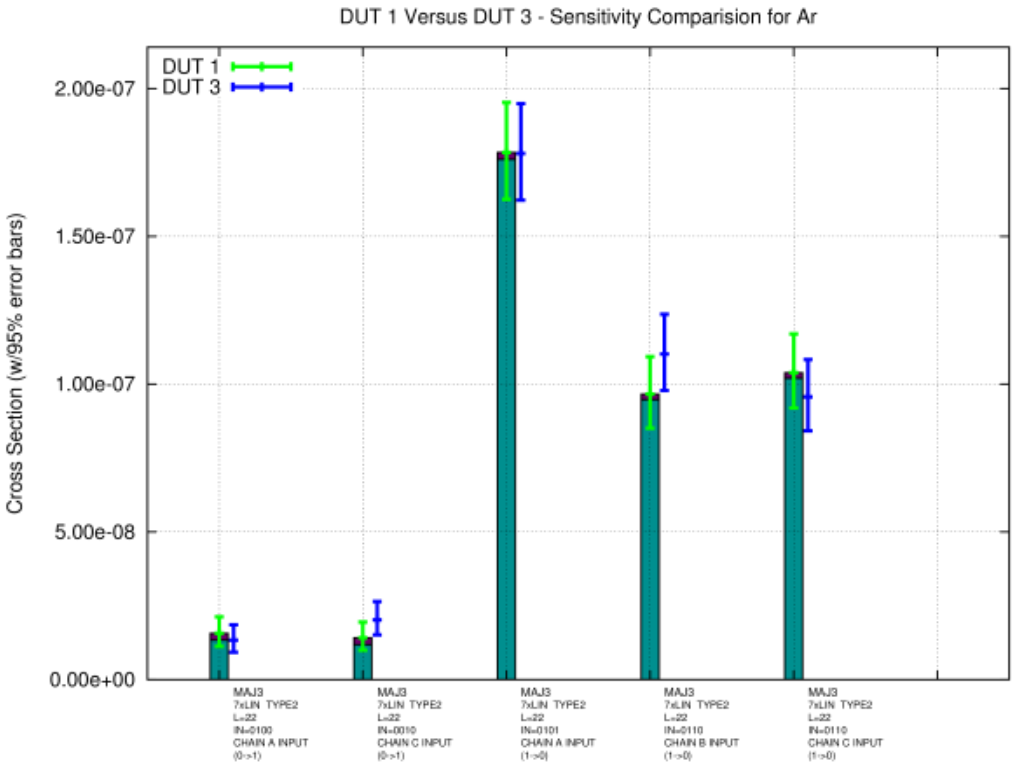

Figure 21 – DUT1,3 Correlation MAJ3 (Ar)

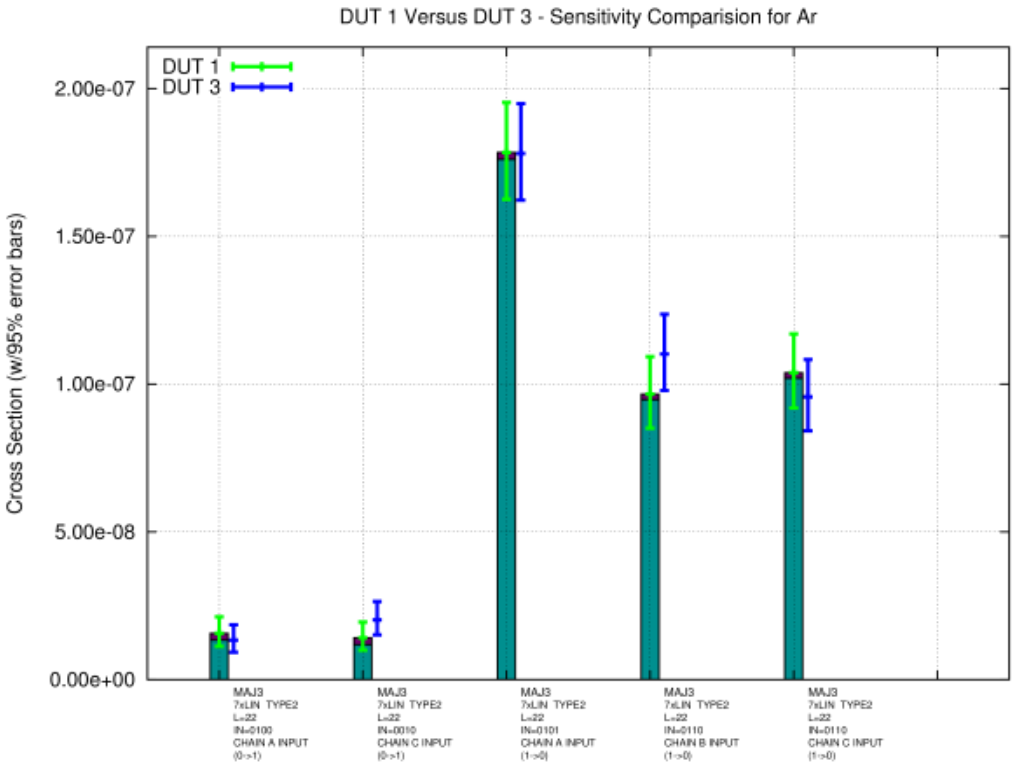

Figure 22 – DUT1,3 Correlation MAJ3(Ne)

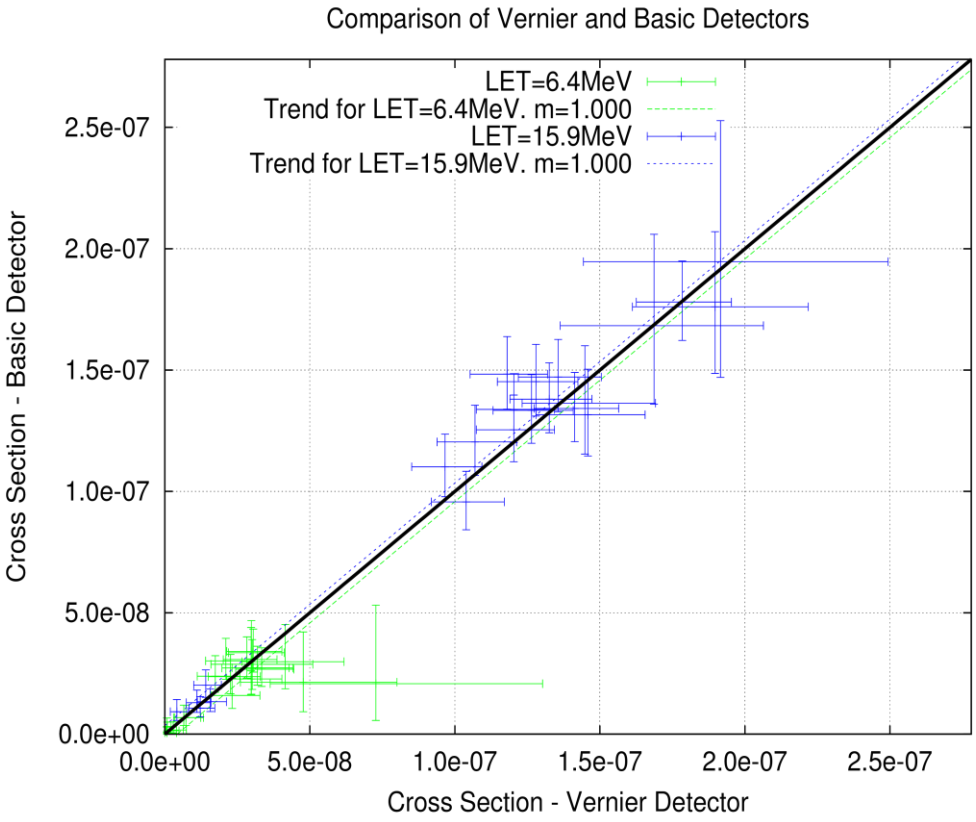

Figure 23 – Correlation of Measured Cross Sections – DUT1 versus DUT3

In order to ensure the consistency and repeatability of the measurements from the Vernier circuit, eight identical detectors (7x22 BUF) were placed on DUT 1 (see green entries in Table 2). The measured pulse width (min/max/average) of these eight detectors is shown in Figure 24 for four different LETs. We observe that the measured average pulse width is very consistent. Generally, the maximum pulse width is consistent; however, one of the detectors shows larger maximum pulse width for higher LETs (Kr, Xe). A possible explanation for the difference in the maximum measured pulse width may be due to difference in the routing for that instance.

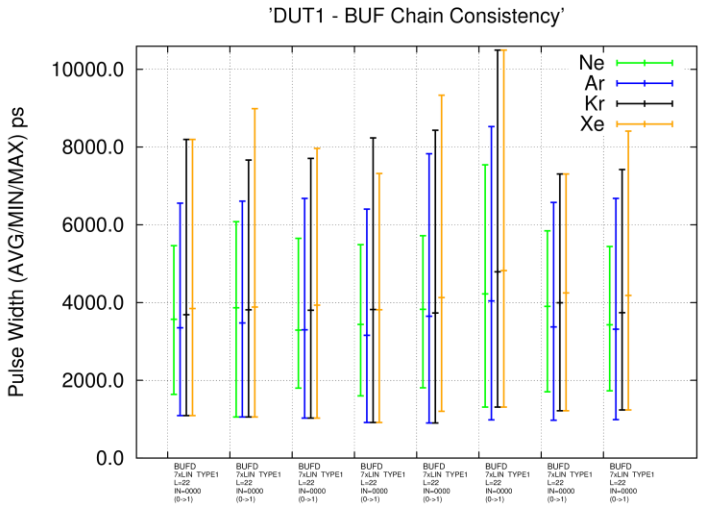

Figure 24 – Consistency of Measured Pulse Width for Identical Detectors

4.3 Average SET Cross Section

The average cross section of all the combinatorial Versatiles in all 92 sensors was calculated at four different LETs and three voltage conditions and these are presented in Figure 25. As expected, at the higher operating voltage (1.5V), the cross section is lower than at the nominal voltage (1.2V).

The measured cross section at the lowest voltage (1.08V) appears to be lower than at the nominal voltage (1.2V). Normally, this is not expected, however, at the lower voltage, the setup and hold times for the flip-flops and latches used in the detectors increases. As a result, the narrower transients, produced at lower LETs, go undetected.

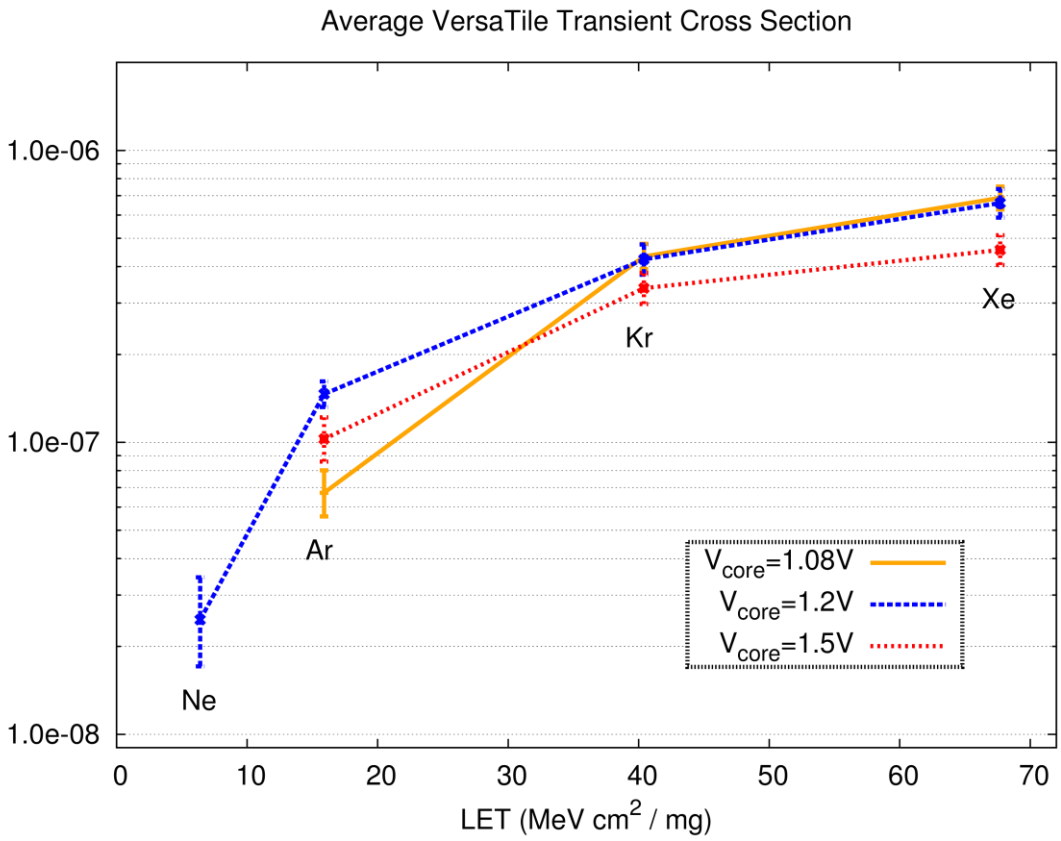

Figure 25 – Average VersaTile Cross Section Versus LET and Voltage

4.4 Input Vector and VersaTile Sensitivity

Using the variety of sensors present in DUT1, the cross section of the numerous different VersaTiles were tested including variations on the input vector state. These results are plotted in Figure 26 through Figure 28. This data shows that the SET cross section for a given VersaTile configuration can vary by an order of magnitude depending on the state of the input signals.

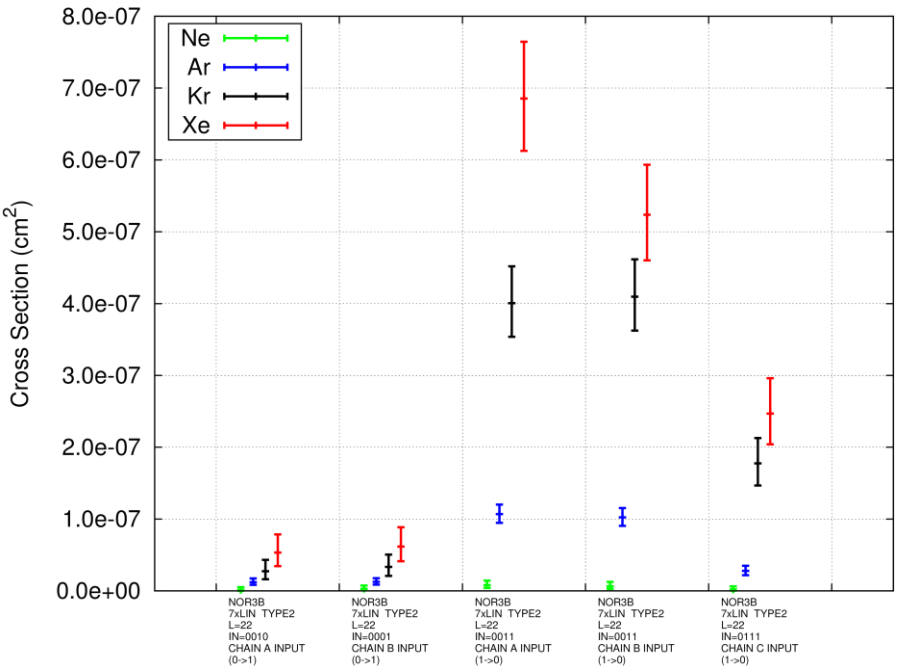

Figure 26 – NOR3B VersaTile Cross Section

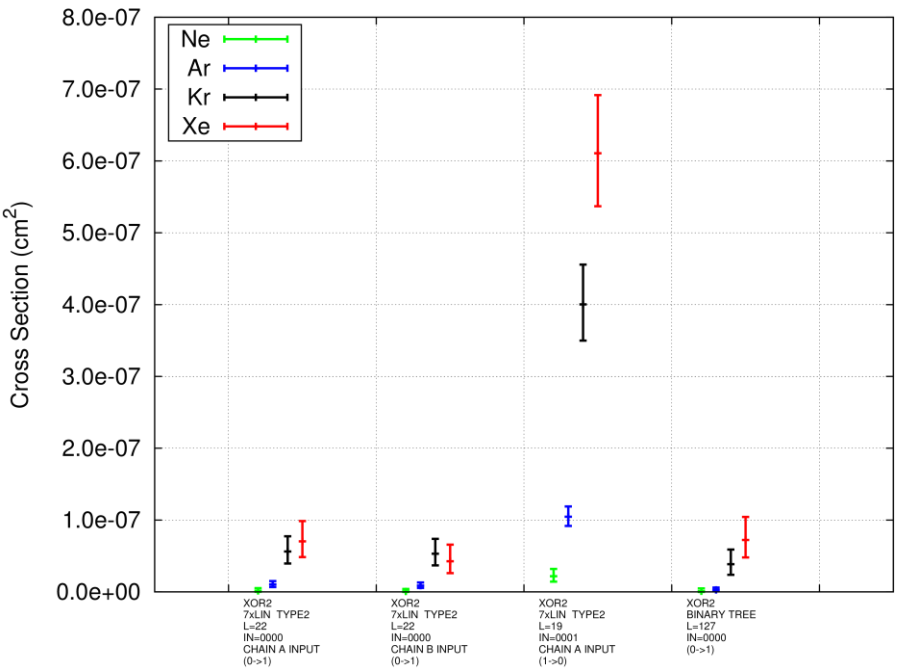

Figure 27 – XOR2 VersaTile Cross Section

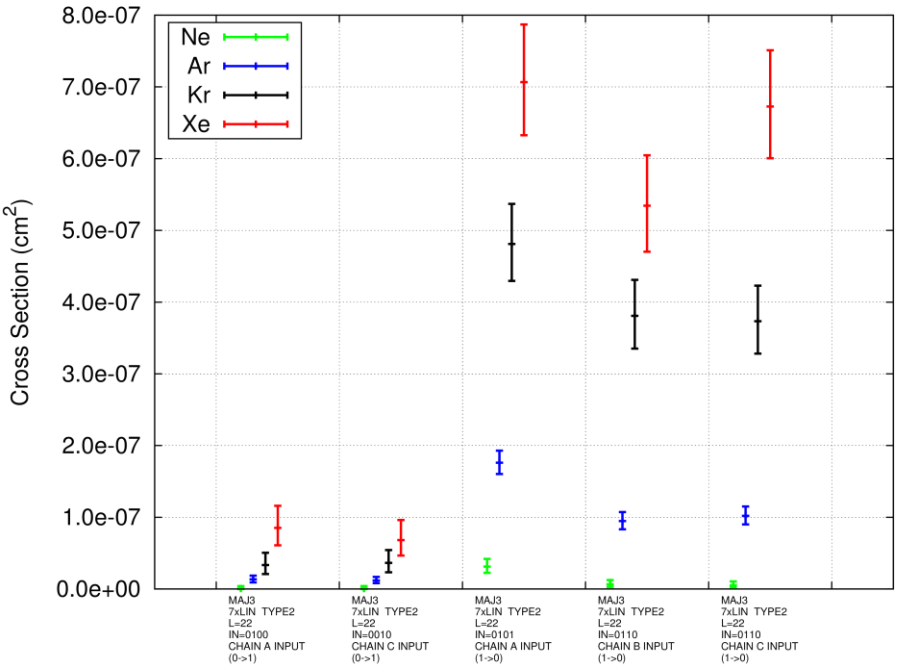

Figure 28 – MAJ3 VersaTile Cross Section

4.5 Effect of Voltage

The DUT configuration with the Basic SET detector was tested at 3 different voltages (1.08V, 1.2V and 1.5V). The average cross section per Versa-Tile, taken across the 92 sensors, is shown in Figure 29. We observe that the difference in SET sensitivity between 1.08V and 1.5V is about a factor of 1.5x.

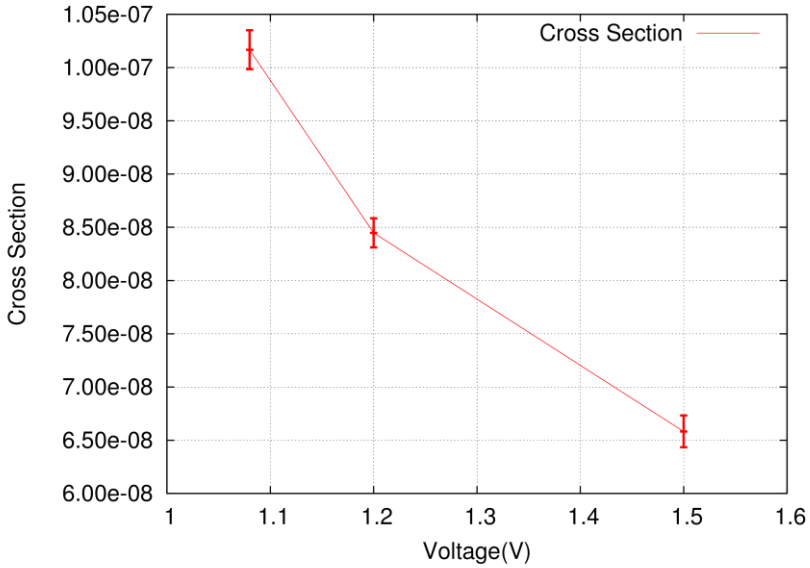

Figure 29 Average VersaTile SET Cross Section Versus Voltage

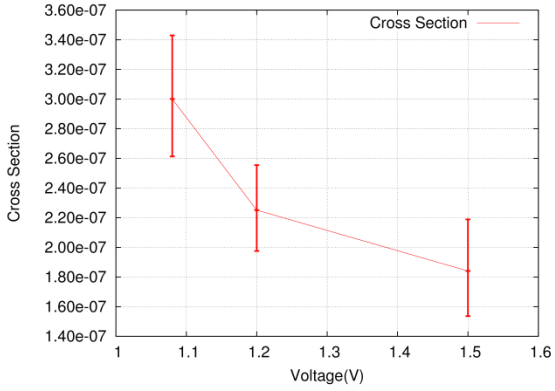

Figure 30 – Cross Section Versus Voltage OR2 VersaTile – Binary Tree (0->1)

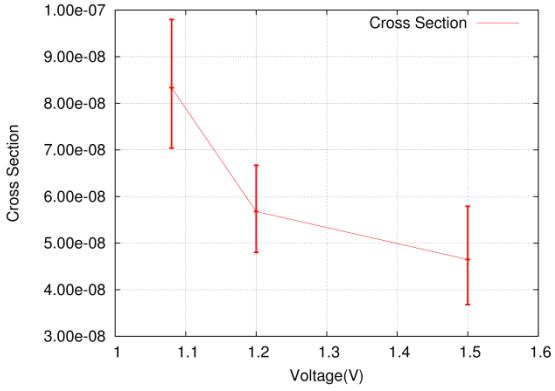

Figure 31 – Cross Section Versus Voltage NOR2A – 7x Linear (Alternating)

When looking at specific sensors (see Figure 30, Figure 31), for positive (0->1) transients, the voltage trend is similar; however, the error bars are larger due to the fact that there are many fewer events.

4.6 Effect of Temperature

The Vernier configuration (DUT1) was tested at three different temperatures (nominal – assumed to be 25°C, 40°C and 70°C). The temperature was controlled by a heating pad on the back side of the package. The temperature was monitored using a thermocouple located at the top of the package and connected to a PID controller.

The runs where temperature was varied were performed at 1.2V. There was no measurable variation of the SET cross section at different temperatures, as shown in Figure 32. This graph plots the average SET cross section versus temperature for all 92 detectors and the variation is within the error bars.

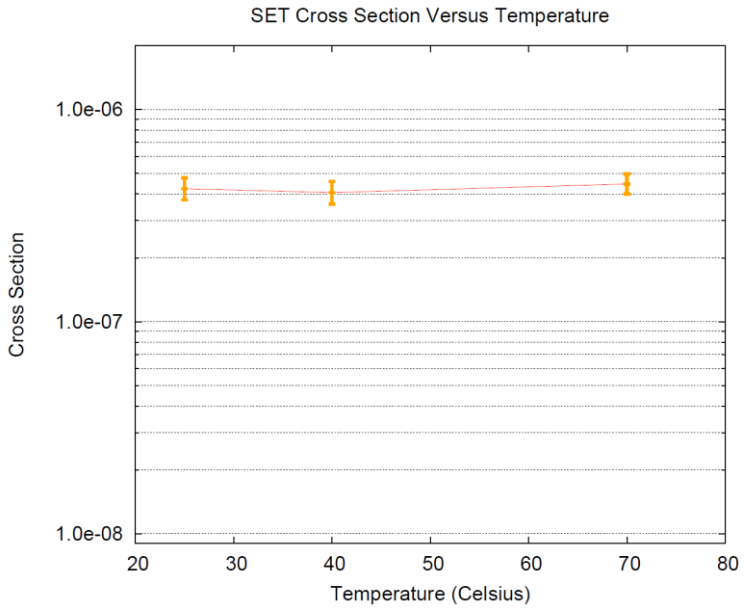

Figure 32 – SET Cross Section versus Temperature (DUT1)

The effect of temperature on the average pulse width was also analyzed. The average pulse width measured from four different detectors is shown in Figure 33 through Figure 36. These graphs show the minimum, average and maximum pulse width measured with the detector at the three different temperatures. It does not appear that there is any specific dependency of the measured pulse width on the temperature.

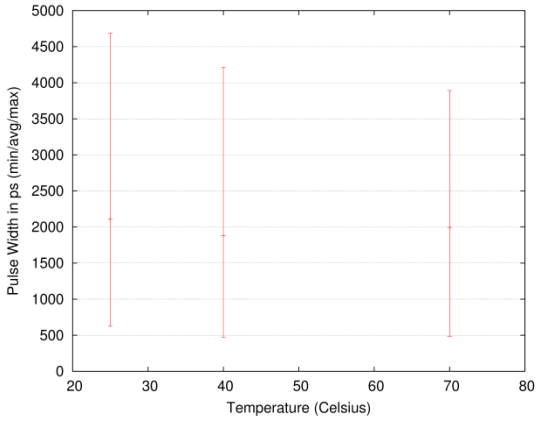

Figure 33 – Avg, Pulse Width vs Temperature (Det. 0, BUF)

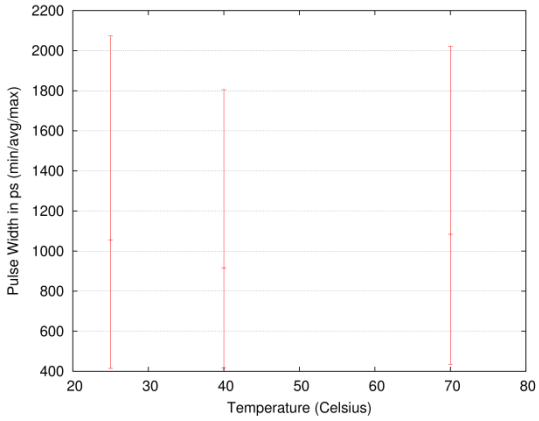

Figure 34 – Avg. Pulse Width vs. Temperature (Det 55, NOR2B)

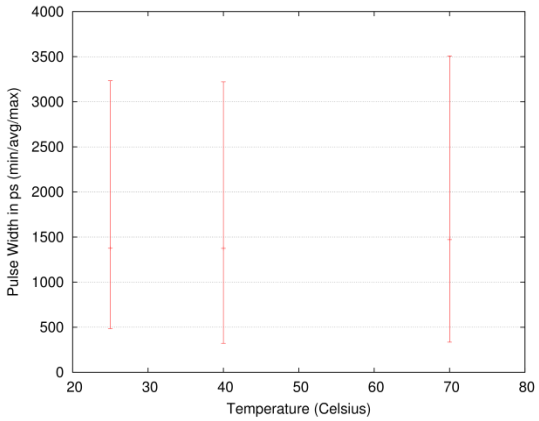

Figure 35 – Avg. Pulse Width vs. Temperature (Det 68, XA1)

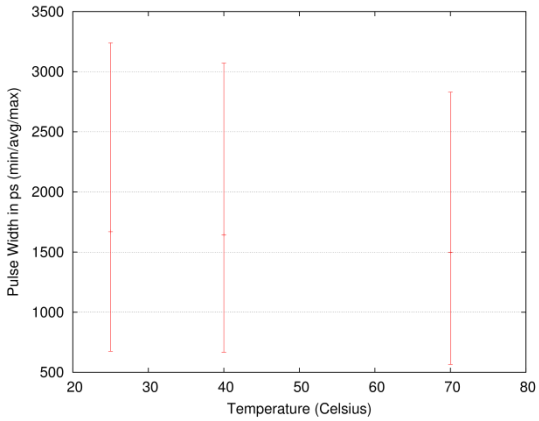

Figure 36 – Avg Pulse Width vs. Temperature (Det.31, OR2)

4.7 Pulse Broadening / Narrowing

Amongst the detectors in DUT1 there were several series of chains of the same gate type with increasing lengths. At the output of such a chain, the observed pulse width is the sum of the intrinsic pulse width distribution of the gate expanded by the broadening/narrowing effect as it propagates along the chain.

In the ProASIC3 technology, there is a very strong broadening effect for positive transients (0->1) as can be seen in **Error! Reference source not found.** and Figure 38.

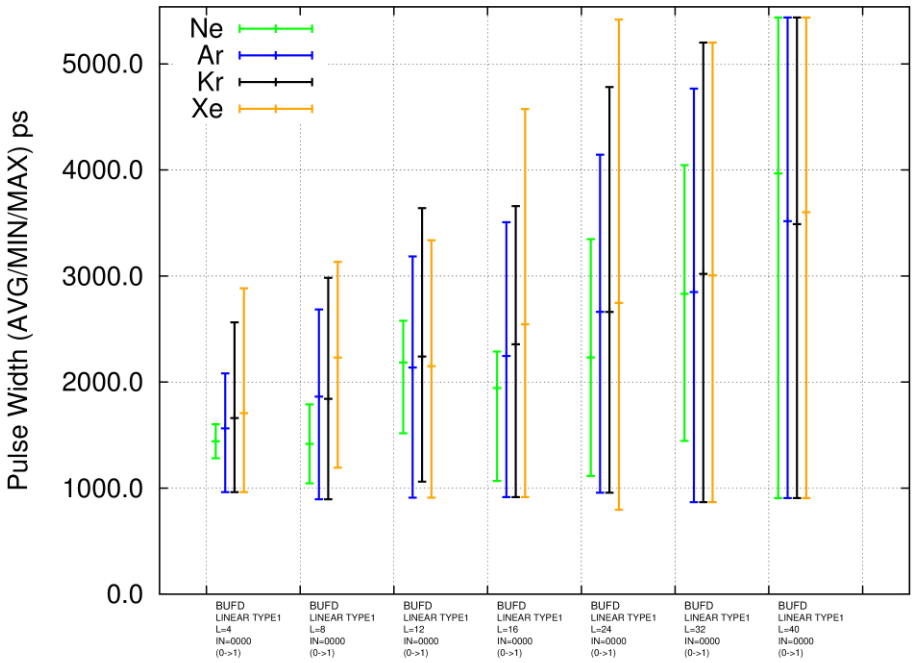

Figure 37 – Pulse Broadening BUF

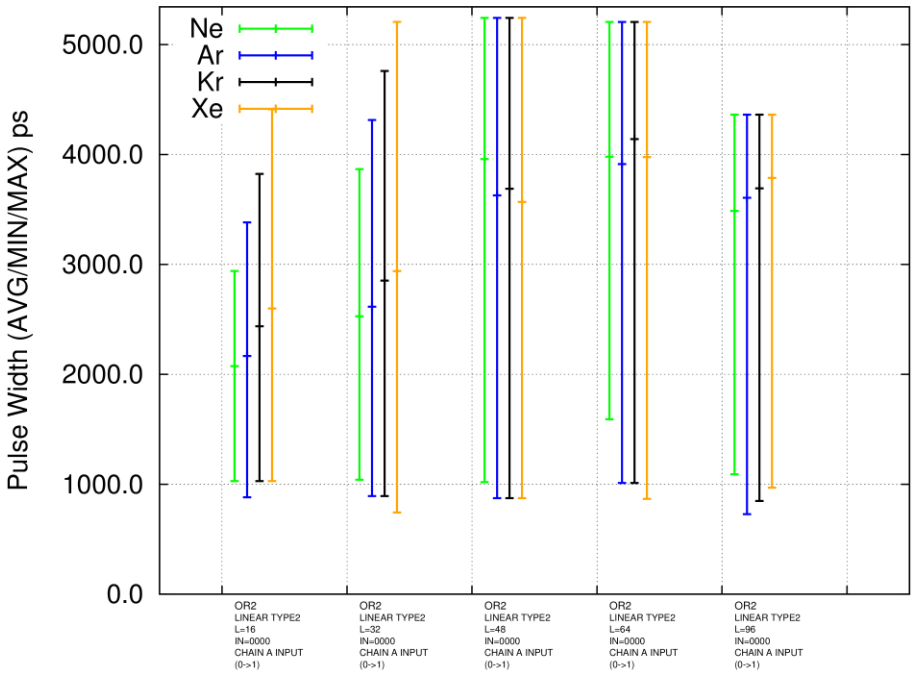

Figure 38 - Pulse Broadening OR2

In order to calculate the per-gate broadening, the maximum measured pulse width was plotted versus the number of stages and a linear fit was performed[1]. For the BUF gate, the result is shown in Figure 39 and the average broadening per BUF per-gate is summarized in Table 6. For the OR2 gate, the results are summarized in Figure 40 and Table 7.

We note that for both gates, the per-gate broadening is essentially independent of the LET. This makes sense as the broadening is a function of the switching characteristics of the circuit.

The intercept of the regression lines, gives an indication of the largest transient that is induced in a single gate and this is also shown in Table 6 and Table 7. We see that the pulse width induced in the OR2 is significantly (30-60%) wider than in the BUF.

[1] For the linear regression, the data for the chain with 40 BUF gates was excluded as the detectors maximum pulse width had been saturated. For the OR2 gate, only the chains with 16 and 32 gates were considered, as beyond this the detector was saturated.

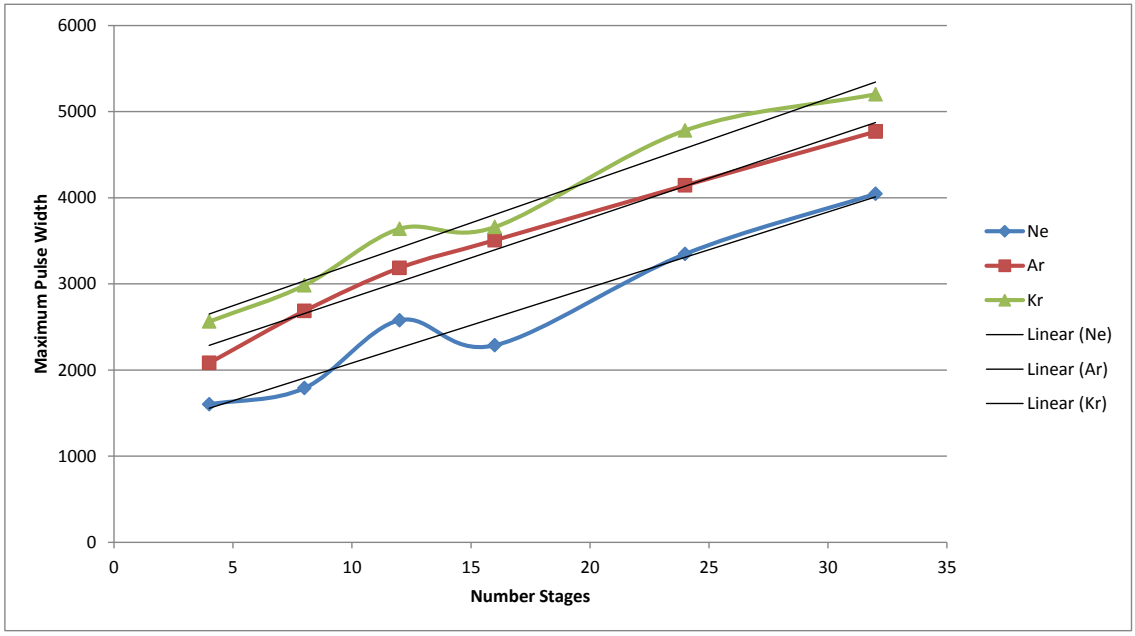

Figure 39 – BUF Gate Broadening

Table 6 – Pulse Broadening and Maximum Pulse Width for BUF Gate

Ion	Slope Broadening Per Gate (ps / gate)	Intercept Maximum Pulse – Single Gate (ps)
Ne (6.4 Mev cm^2 /mg)	87	1207
Ar (15.9 Mev cm^2 /mg)	92	1918
Kr (40.4 Mev cm^2 /mg)	96	2267

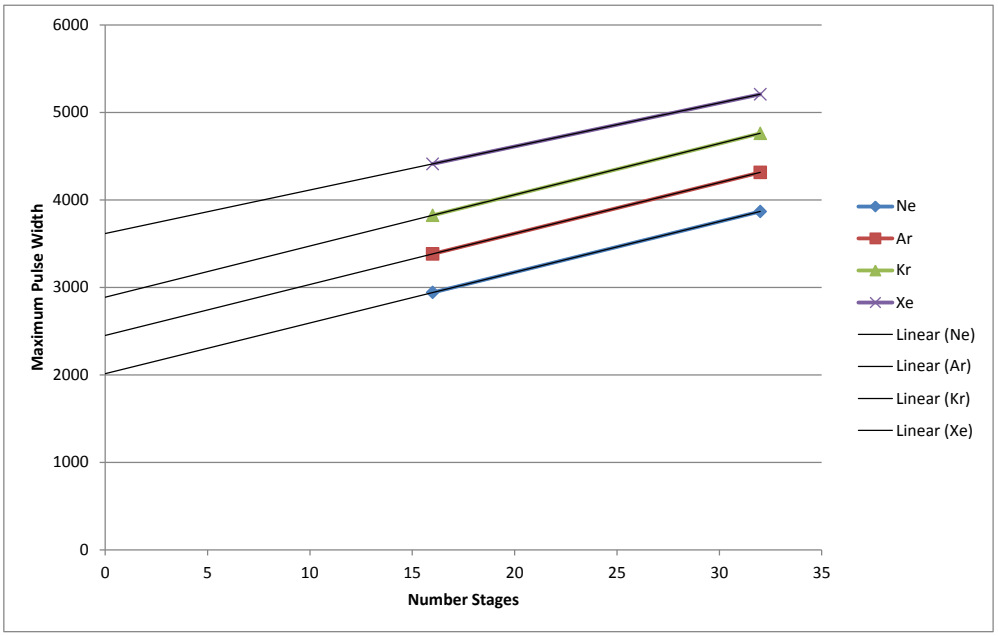

Figure 40 – OR2 Pulse Broadening

Table 7 – Pulse Broadening and Maximum Pulse Width for OR2 Gate

Ion	Slope Broadening Per Gate (ps / gate)	Intercept Maximum Pulse – Single Gate (ps)
Ne (6.4 Mev cm^2 /mg)	58	2014
Ar (15.9 Mev cm^2 /mg)	58	2450
Kr (40.4 Mev cm^2 /mg)	59	2886
Xe (67.7 Mev cm^2 /mg)	50	3616

4.8 SET Measurements in Complex Circuits

Using the structure shown in Figure 12, the seven complex circuits listed in Table 3 were tested. It is important to note that the test structure makes it possible to clearly differentiate between SETs in the combinatorial logic and SEUs in the surrounding flip-flops. The cross section of SETs in the complex circuits is plotted in Figure 41. The cross section is normalized based on the number of Versatiles (e.g. gates) in the circuit.

We observe that the ECC encoder has the highest cross section, which is expected since there is absolutely no logical masking (all XOR). The priority encoder and the mux have lower cross sections, since these circuits intrinsically have more logical masking. In a mux, upsets that do not affect the selected output do not propagate. A priority encoder indicates the highest numbered bit

in the input vector (see Appendix B), thus transients that occur in bits related to lower numbered bits are likely to be masked.

Of course, the cross-section of gates in complex circuits is much lower than in isolation due to logical and temporal masking effects. The circuits were operated at 10 MHz and the average intrinsic measured pulse width was between 1.2..2.8 ns (depending on the type of cell and the state, see Section 4.7). This gives a temporal de-rating factor between 12% and 28%.

In Table 8, the typical SET cross-section of combinatorial VersaTiles measured using the static sensors and measured in dynamic circuits is compared. For lower LETs, the pulses are narrower and there is thus more temporal de-rating, as expected.

Overall, it appears that the effective contribution of SETs in complex circuits is two orders of magnitude lower than their static rate of occurrence. A factor of 4x to 8x may be explained by temporal masking. The remaining masking can not be explained purely by logical masking; for example an ECC encoder has no logical masking. Some of the additional masking may come from the very strong pulse narrowing that was observed for negative (1->0) transients.

Table 8 – Comparison of Static and Dynamic Cross Section Measurements

LET	**Typical VersaTile Cross Section**	**Typical Complex Circuit Cross Section**	**Ratio of Complex to Basic**
Ne (6.4 Mev cm^2 /mg)	2e-8	5e-11	400
Ar (15.9 Mev cm^2 /mg)	2e-7	1e-9	200
Kr (40.4 Mev cm^2 /mg)	4e-7	1e-8	40

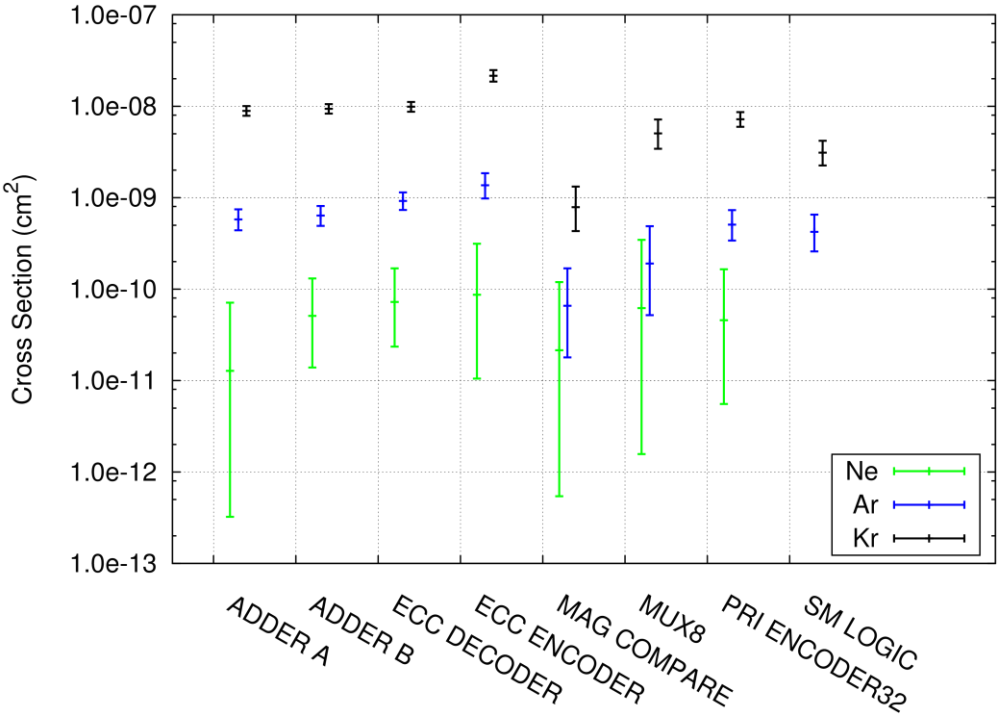

Figure 41 – SET Cross Section of Complex Circuits (per VersaTile)

When a transient occurs in a complex circuit, it may affect multiple output bits, depending on the fanout of the logic. The graph in Figure 42 shows a plot of the number of output bits that were upsets, for each complex circuit. The error bars show the smallest and the largest number of upset bits that were observed.

From this graph, it is clear that the majority of the transients affect only one bit. At higher LETs, however, multiple bits can be upset, with some transients affecting up to 6 output bits. This observation is important, as techniques such as SEC ECC and do not provide protection against these faults.

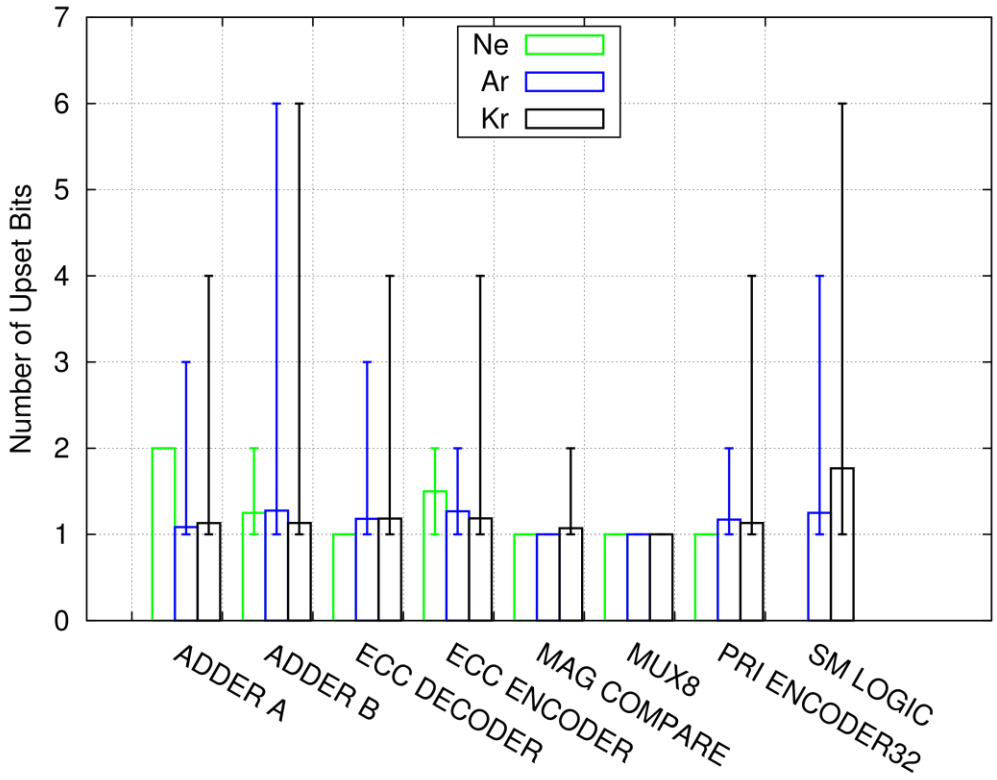

Figure 42 – Number of Output Bits Upset

4.9 Permanent Effects

During the HI testing at UCL, permanent effects were observed on the devices under test. Between test runs, it was necessary to re-program the FPGAs to implement the different DUTs.

The reconfiguration was always performed when the beam was off using the standard FlashPro Software and FlashPro4 programmer from Microsemi. Verification failures (such as "Verify 0 failed at row 7843") were observed very early in the test campaign. Specifically, one of the devices failed during the first experiment run, after a few minutes of exposure to a high flux (1.5e4 part/sq. cm/s) of Xenon ions (LET = 67.7 MeV/mg cm^2). This represents a total dose of 4.2KRad:

$$Dose(in\ Rad) = 1.602e-5 \times Fluence\left(\frac{part}{cm^2}\right) \times LET\left(\frac{MeV}{mg \circ cm^2}\right)$$

$$4.3\ Krad = 1.62e-5 * 3931372 * 67.7$$

The programming procedure (using the standard, binary .pdb files generated by Libero) halts in the case of a verification error. The FPGA is not enabled and is thus not functional.

In order to continue testing, an alternative method to program the device was used. The Libero software can generate a text file format (.STAPL). This text file can be manipulated in a text editor to suppress the verification phase and force the device to be enabled.

Using this procedure the FPGAs were successfully re-programmed and appeared to function correctly. For example, the registers read/write operations functioned correctly (see Section 3.1) and the delay calibration sequence executed correctly (see Section 3.2.2). Interestingly, verifying the FPGA content after programming, using the standalone verify procedure was also successful.

We note that other work has identified similar problems in the ProASIC FPGA family. In RD16, the authors report similar error messages were observed after exposure to 50-60KRad. In RD18, programming failures were reported between 10KRad and 28KRad of proton beam radiation.

We note that in Micro Semi's Dose Report (RD15), they do not appear to test whether devices can be re-programmed after exposure.

4.9.1 Recovery from Permanent Effects

The two damaged FPGAs were monitored during the months subsequent to the testing. For a period of six weeks, the FPGAs were stored at room temperature. Daily reprogramming procedures (i.e. including the verification step) were performed, but no improvement was observed; verification errors were reported at each attempt.

Subsequently, the devices were annealed by heating them to 100°C. After 4 days of annealing, one device was able to be re-programmed correctly, while the other showed sporadic errors (40% of the reprogramming attempts succeeded). The annealing process was continued for an additional 3 weeks and no further improvement was observed for the malfunctioning device.

Finally, the device was left at room temperature for approximately 4 months. After this time, we were able to repeatedly program and configure the device correctly without no discernible functional problems.

For comparison, in RD16, it was reported that after 3 weeks of annealing at room temperature, the devices could still not be re-programmed.

5 Laser Test Results

On September 13-14, 2013, the FPGA devices were tested under a pulsed laser at EADS Innovation Works (Paris).

During this test-campaign, it was possible to validate the operation of the Vernier SET detector and to obtain some limited measurements. Due to some technical issues, that are detailed in Section 5.3, the extent of the measurements are limited.

5.1 Energy Versus Pulse Width

A series of runs were performed with different laser pulse energies (1nJ, 1.5nJ, 3nJ, 4.5nJ) and at three different voltages (1.08V, 1.2V, 1.5V). For each run, the laser was swept over a 10umX10um region covering a BUF VersaTile near the start of the chain for detector #0. Unfortunately, we do not have the exact location of the Versatile, so the measured results may be subject to some pulse broadening.

The results have been plotted in Figure 43. For the 1.08V runs, there is a clear, linear trend between the laser energy and the induced pulse width. For the higher voltages, the Vernier detector saturates (note – the chain length was 48 stages). The reason it saturates at higher voltages, is that at higher voltages, the device is faster, thus the same number of stages represents a shorter absolute delay. However, in the region of 1nJ..3nJ, the linear relationship is observed.

The effect of the voltage on the pulse width is interesting, and not necessarily what would be expected.

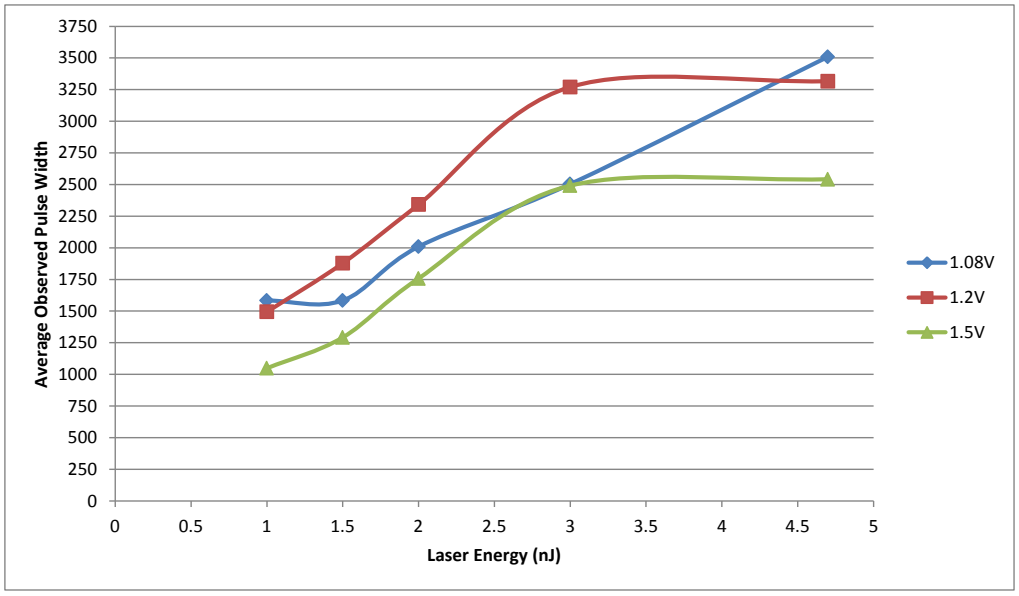

Figure 43 – Measured Pulse Width versus Laser Energy

5.2 Cartography of Sensitivity

Due to the fact there was no mechanical synchronization, after the testing, it was difficult to correlate the events reported in the log file with the location where the laser was fired. However, using the time-stamps of the events, with significant effort, we attempted to work backwards to identify the location. The difficulty lies from the fact that the log file only contains the events that produced events. When there is a long temporal gap with no events in the log file, it is difficult to accurately situate the first event after the gap. An algorithm was developed which analyzed the frequency of inter-event gaps to identify the vertical lines and then to identify the horizontal position within a line.

The results of using the algorithm to study the sensitivity of the BUF gate (SET sensor #0) are shown in Figure 44 through Figure 47 for four different laser energies. The colour of the dots gives a relative indication of the width of the transients. We note that there is a correlation in the sensitive region between the four images and clearly, the higher energies produce a greater sensitivity and wider transients.

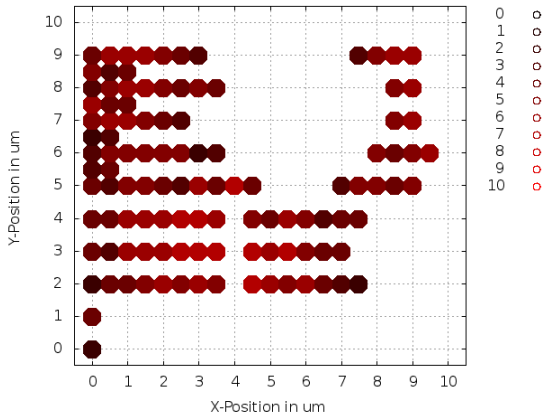

Figure 44 – BUF VersaTile Image (4.7 nJ)

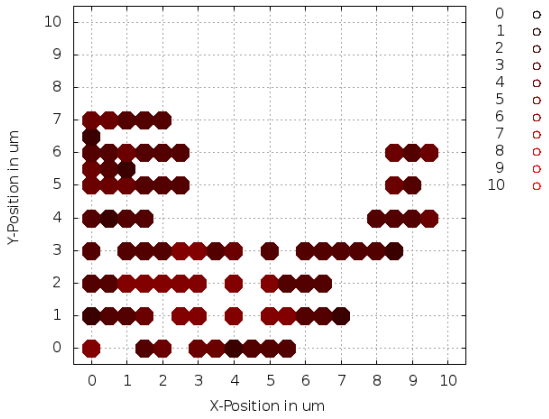

Figure 45 – BUF VersaTile Image (3.0 nJ)

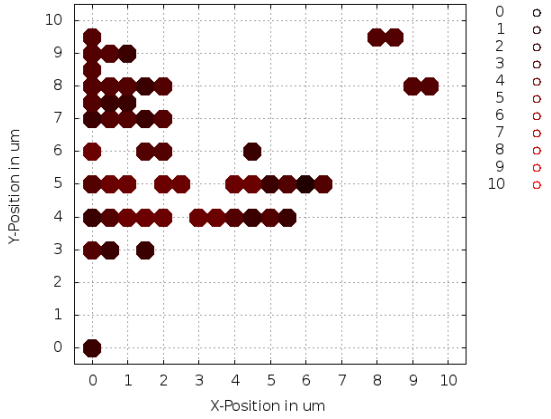

Figure 46 – BUF VersaTile Image (2.0 nJ)

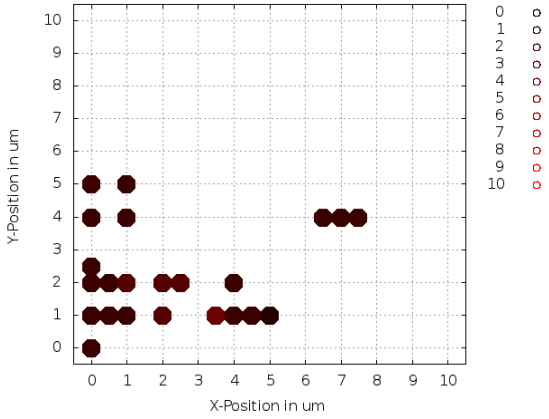

Figure 47 - BUF VersaTile Image (1.5 nJ)

5.3 Issues Encountered During Laser Testing

During the laser test campaign, there were some issues with the alignment of the DUT and with the synchronization with the laser.

5.3.1 Angular Alignment

An issue that became apparent during the initial calibration was that angular alignment of the DUT was not parallel/perpindicular to the axes of the motor bed (X/Y). As a result, when a sweep was performed, with the intent of running the laser along a specific row or column of Versatiles, the laser was in fact drifting into adjacent rows/columns (see Figure 48). Even when the laser was staying within a row of versatiles, it was not always hitting the same transistors within the Versatile.

As a result, during some sweeps, it was observed that certain SET sensors in the DUT would fire, but others that were further along would not fire, because the laser had drifted out of the sensitive regions.

The first step to resolve this issue is to use a test setup for the board that is rigid. This alone may not be adequate because the die itself may not be aligned within the package and the package may not be perfectly aligned on the board. A second step is to use a adjustable table that enables fine angular adjustment of the DUT. Finally, having improved synchronization makes it possible to know the exact location of all shots fired by the laser. In this way, it is possible to compensate for minor alignment issues in the post-processing.

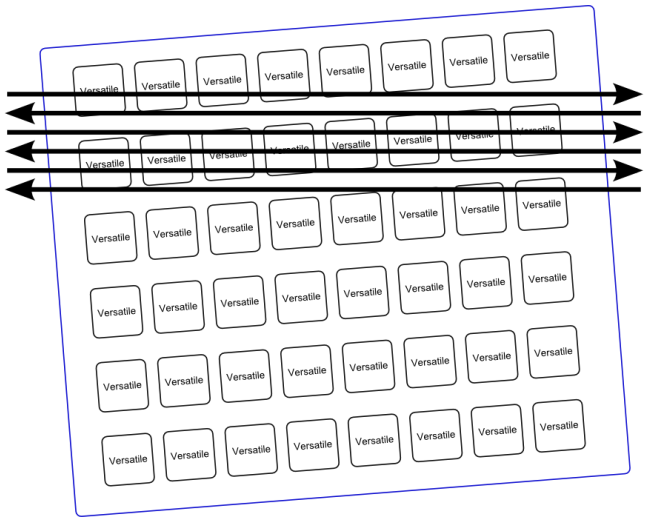

Figure 48 – Alignment Issue with Mechanical Stage

5.3.2 Planar Alignment

The focus of the laser is very important and it was clearly observed that when the laser was not correctly focused at the correct depth within the substrate, no upsets were produced, even with high energy levels. To maintain the focus across the entire die (approximately 1cm) requires tight mechanical tolerances.

The first order problem with the initial test-setup was the tolerances in the jig used to support the test card (see Figure 49). The circuit board that was used as a mechanical support had some play and there was some play in the vertical spacers. As a result, the die was not level and the laser focus could only be maintained for sweep distances of less than 1mm.

The first order issues were fixed on-site : the support was re-drilled to improve the alignment on the motor bed platform and an adjustable table was added to the setup.

After the mechanical setup was improved, it was observed that the laser focus could be maintained for sweep distances of approximately 1-2 mm. However, it was still difficult to maintain the focus over the entire surface of the die (1cmx1cm). The remaining non-planarity was likely the result of the die surface being slightly bowed.

The first order planarity issues can be resolved with a more rigid test support and the use of an adjustable support table. Since this is not necessarily sufficient to ensure a good focus across the entire die surface, it is important to design the experiments in the test-plan such that it is not necessary to sweep the entire die at one time. Instead, it is preferable to identify multiple experiments which can be performed over surfaces of approximately 1-4 mm2. Over a smaller area, it is much easier to ensure the laser remains focused and ensure that the elements being tested receive a constant level of injected energy.

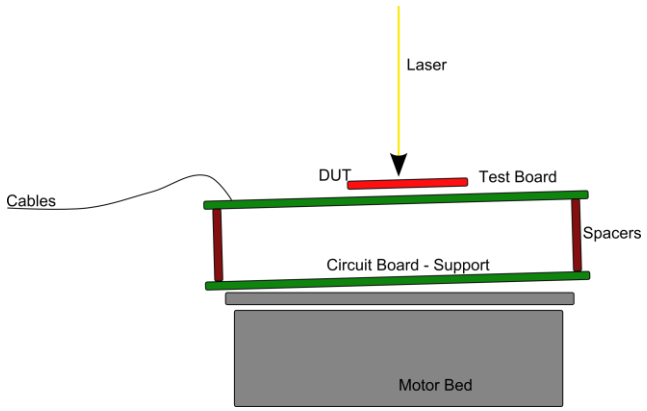

Figure 49 – Planar Alignment

5.3.3 Improved Setup

For any future laser or micro-beam testing, it is critical to have a tight synchronization between the recorded events and the location of the laser or beam. One approach to improved synchronization is shown in Figure 50. With this approach, each time the laser fires, it causes a special event to be sent to the IROC tester resulting in a record in the log file. In this way, all the shots are recorded, including those that produce no error. In addition, the response of the DUT (error or no error) is sent back to a digital scope which can record the event.

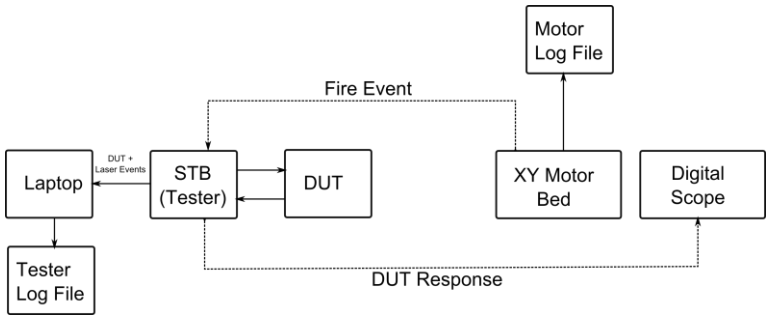

Figure 50 – Improved Synchronization

6 Conclusions

Accurately measuring SETs in FPGAs is challenging. First lays the difficulty of observing the transients. Since the FPGA hardware is fixed and the IOs have relatively low bandwidth, the observations must be done on-chip. The Vernier sensor that was developed in this project has been demonstrated to have a good detection capability through extensive comparison of the measured cross sections with a basic detector. It was also effective at measuring the width of the transients. However, when the input to the detector is a chain of gates, what is measured is a distribution of the original pulses that have been subjected to broadening or narrowing as they propagate along the chain. This complicates the measurement of the width of the pulses that were originally induced by the SET, which must be determined by extrapolation.

The difficulty of the problem of characterizing SETs is compounded by the fact that the sensitivity depends very significantly on how the VersaTile is configured and on the state of the input signals. Furthermore, in the ProASIC3 technology, positive (0->1) transients are subject to pulse broadening and negative transients (1->0) are subject to significant narrowing and thus suppression.

What is of most importance is to characterize the rate of errors produced by SETs in typical circuits. The study of SETs in seven complex circuits shows that only a small fraction of SETs are captured as errors. Transients can produce upsets in multiple output bits in a complex circuit, and this must be considered when using parity or SEC error correction.

We enumerate the major findings of this study:

- The SET cross section of a single ProASIC3 VersaTile is on the order of 1e-7 however this can vary by an order of magnitude depending on the VersaTile configuration and the state of the inputs.
- By extrapolation, the maximum intrinsic SET pulse width can vary from 1.2ns (BUF VersaTile, LET= 6.4 MeV/mg cm^2) up to 3.6ns (OR2 VersaTile, LET=67.7 MeV/mg cm^2).
- The average SET cross section is approximately 1.5X higher at 1.08V than at 1.5V
- The SET cross section and the SET pulse width have very little sensitivity to temperature
- Positive SETs (0->1) are subject to significant broadening (90ps/gate for BUF, 60ps/gate for OR2). For example, 10 stages of logic could result in a broadening of close to 1ns.
- Negative SETs (1->0) are subject to a very significant narrowing, although it is difficult to quantify this effect because transients occurring in long chains are mostly suppressed.
- The SET cross section of VersaTiles used in complex circuits (operation at 10MHz) is significantly lower than the statically measured SET sensitivity (on the order to 1e-9 per VersaTile). This gap is approximately two orders of magnitude and is the result of temporal and logical masking as well as pulse narrowing.
- Destructive effects were observed on multiple devices. The symptom was that the device could not be re-programmed, although it did appear to continue to function correctly.

6.1 Estimating Mission Event Rates

In order to better understand the impact of transients, we have computed the estimated failure rates due to SETs induced by heavy-ions for actual clocked circuits based on the following assumptions:

- MicroSemi ProASIC3 Device with 25 000 combinatorial VersaTiles
- Logic Operating at 10 MHz (same as measurements of complex circuits)
- 10 year mission in Geo Stationary Orbit (GEO)

LET-dependent cross-section data is a required input for any calculations aiming at providing applicative, mission-specific event and failure rates. The pre-requisites for this analysis include: intrinsic, cross-section data as a function of the LET of the particles, FPGA configuration information and particle flux distribution for the considered environment.

The works presented in the previous chapters provided cross-section data for a limited selection of heavy ions species in actual clocked circuits. Thus, we have extrapolated the available data to match a wide LET spectrum spreading from 1.7e-3 MeV.cm^2/mg up to 100 MeV.cm^2/mg. The cross-section associated to the LET of the highest particle tested has been selected as saturation cross-section. The interpolated curve is presented in Figure 51 as well as the actual observed cross-section for three of the complex combinatorial circuits.

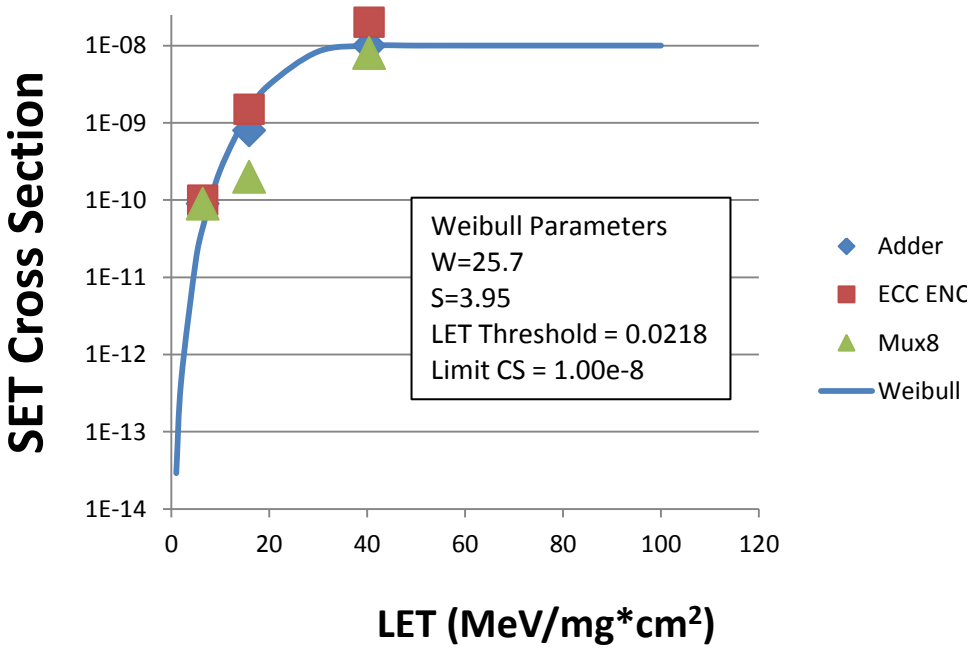

Figure 51 – Weibull FIT to Effective SET Cross Section of Complex Circuits

Furthermore, we have used the OMERE toolkit [RD19] to generate the flux distribution of heavy ions coming from Galactic Cosmic Rays and solar activity, according to the following main assumptions: GCR heavy ions' (from H to U) flux distribution vs LET match the GCR ISO 15390 standard, solar heavy ions use the PSYCHIC model (He to U) and lastly an Al shielding of 3.705 mm has been applied.

Finally, the interpolated CS(LET) data and the Flux(LET) information made possible the computation of an event rate per VersaTile and per time period. Assuming that 25 000 VersaTiles are used in the case of a realistic system based on the considered FPGA, we can compute an overall device event rate per day or mission for heavy ions. Table 9 presents the most important findings.

Table 9 – Effective Event Rates for a Large FPGA

Events / Second	1.8E-08
Events / Day	0.0016
Events / 10 years	5.8

We note that in general terms, this event rate will generally scale linearly with frequency, so at 100 MHz, the rate would be 10x higher. We also note that all the low-level SET de-ratings have been taken into account (electrical, logical and temporal masking).

For comparison, if the SEU cross sections reported in [RD3] are considered, and based on the same mission parameters, the event rate for SEUs would be approximately 0.1 per day (for 25 000 VersaTile flip-flops). Thus at 10 MHz, the effective SET contribution is approximately 1.5% of the SEU sensitivity. However, at 100 MHz, this ratio would be closer to 15%.

Finally and more importantly, the rate calculated above is estimated from the selection of clocked circuit examples measured in this work. Since there is significant variability depending on the actual circuit function, the SET rate of the MicroSemi ProASIC3 Device should be calculated with the exact programmed function used in the flight application.

6.2 Perspectives for Future Work

Overall, the Vernier SET detector was effective at measuring the transients that were generated in various test structures. Of course, the detector can only measure the pulses at the output of a sensor built from combinatorial gates. Great care must be taken in selecting the topology of the collectors. Although long chains are useful in order to increase the total sensitive area, they result in a significant distortion of the pulse width. In this work, the pulse width in individual gates was determined by measuring the pulse broadening from a series of chains of different lengths, and then extrapolating this trend back to the expected pulse width for a single gate.

In the case of negative transients which are largely suppressed due to narrowing, increasing the length of the chain does not increase the sensitive area, thus making them difficult to measure.

The measurements performed on the complex circuits showed that there is a significant gap between the static SET rate observed using detectors and the effective SET rate when the gates

are used in typical circuits. This gap is the result of temporal, electrical and logical de-rating. These factors could have been better studied if multiple test-runs had been performed at different operating frequencies. The frequency dependent variation comes from temporal de-rating .

It would be valuable to better decode the error messages that are reported when the devices can not be successfully re-programmed due to dose effects. Specifically, it would be extremely valuable to correlate the bit that can not be re-programmed and the logic-resource that is not operational. Typically, only a small fraction of configuration bits actually control user logic. If MicroSemi reported the Critical bits for a given application, then the user could assess whether the device can reasonably be accepted to operate correctly despite a mis-match during configuration.

Appendix A - HDL Source Code of Vernier SET Monitor

```
////////////////////////////////////////////////////////////////////////////////////
// Company: iROC Technologies (for ESA project AO7236)
//
// File: VernierSETMonitor.v
//
// Description:
//
// This file contains a SET measruement circuit. The pulse to be measured
// arrives on the input 'pulse_in'. The rising edge of the pulse triggers
// a 'start latch' which causes an edge to propagate along chain 1. The
// falling edge of the pulse triggers a 'stop latch' which causes an edge
// to propagate along 'chain 2', which has shorter delays.
//
// After a transient is detected, the output 'int_out' is asserted and the
// output bus 'data_out' holds the binary recording of the transient. Note
// that bit #0 holds the start bit and bit #1 holds the stop bit. The
// remaining bits hold the contents of the latches.
//
// After the transient event has been recorded, the 'reset_local' input must
// be asserted in order to arm the sensor for the next event.
//
// Configuration Parameters:
//
// N_STAGES = Number of stages in the delay chains
// DEFAULT_VALUE = Determines whether sensor detects 0=(0->1) or 1=(1->0) transients
// NMR = Controls whether the reset is duplicated (2) or triplicated (3)
// OUTREG_EWIDTH
//
//
// Author: Adrian Evans (adrian@iroctech.com)
//
////////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//`define DEBUG_ONLY 1

module VernierSETMonitor(  clk,              // clock for control logic
                           reset_global,     // chip-level reset
                           reset_local,      // reset for this sensor

                           int_out,          // an event has been detected
                           data_out,         // contents of capture register

                           chain_1_loop_en,  // Enable chain 1 as a ring-oscillator
                           chain_1_out,      // Observe output of chain 1

                           chain_2_loop_en,  // Enable chain 2 as a ring-oscillator
                           chain_2_out,      // Observe output of chain 2

`ifdef DEBUG_ONLY
                           vernier_chain_1,
                           vernier_chain_2,
`endif

                           pulse_in ) ;       // incoming pulse to be measured,

   // The number of stages in the vernier chain is controlled by the parameter N_STAGES
   parameter N_STAGES = 24;
   parameter NMR = 2;
   parameter DEFAULT_VALUE = 0;    // When default value=0 ---> expect to get a SET that
goes 0->1->1
                                   // When default value=1 ---> expect to get a SET that
goes 1->0->1

   parameter OUT_REG_WIDTH = 64;

   input                       clk;
```

```
   input [NMR-1:0]             reset_global;
   input                       reset_local;
   output                      int_out;
   output [OUT_REG_WIDTH-1:0]  data_out;         // Bit 0 = Start, Bit 1 = Stop,
Bits[N_STAGES+1:2] = Vernier Capture

   input                       chain_1_loop_en;  // 0=Delay line, 1=Enables chain 1 as a
ring-oscillator
   output                      chain_1_out;

   input                       chain_2_loop_en;  // 0=Delay line, 1=Enables chain 1 as a
ring-oscillator
   output                      chain_2_out;

`ifdef DEBUG_ONLY
   output [N_STAGES:0]      vernier_chain_1;
   output [N_STAGES:0]      vernier_chain_2;
`endif

   input                    pulse_in;

    // Declare Signals
   wire                     start;
   wire                     fall_edge;
   wire                     stop;
   wire [N_STAGES:0]        vernier_chain_1 ;
   wire [N_STAGES:0]        vernier_chain_2;
   wire                     chain_2_start;
   wire                     reset_local_FF;
   wire                     reset_local_next;
   reg                      int_out;

   // Flop the reset locally - so the fanout is not too large
   //    Reset on either reset_global==2'b11 -OR- reset_local
   AO1  U_reset_combo_AO(  .A( reset_global[0]  ),
                           .B( reset_global[1]  ),
                           .C( reset_local      ),
                           .Y( reset_local_next ) );

   DFN1 U_reset_local_FF( .D( reset_local_next ),
                        .CLK( clk              ),
                          .Q( reset_local_FF   ) );

   // Start Signal
   // ~~~~~~~~~~~~
   // Start signal is triggered by the leading edge of the SET pulse.
   // Start signal is cleared by reset.
   // Implementation is an active high latch
   generate
      if ( DEFAULT_VALUE == 0 ) begin : START_LATCH_0
         DLN1C1 start_latch( .CLR( pulse_in       ),     // clear is active high
                             .D( 1'b1             ),     // only used for clear
                             .G( reset_local_FF ),     // loads in ine
                             .Q( start            )  );
      end // if
      if ( DEFAULT_VALUE == 1) begin : START_LATCH_1
         DLN1C0 start_latch( .CLR( pulse_in       ),     // clear is active low
                             .D( 1'b1             ),     // only used for clear
                             .G( reset_local_FF ),     // loads in ine
                             .Q( start            )  );
      end // if
   endgenerate

   // Stop Signal
   // ~~~~~~~~~~~
   // Stop signal is triggered on trailing edge of the SET pulse
   // Stop signal is cleared by reset

   generate
      if ( DEFAULT_VALUE == 0 ) begin : STOP_LATCH_0
         DFN0C1 stop_latch( .D( 1'b1           ),
```

```
                          .CLK( pulse_in       ),    // falling-edge triggered - end of
pulse
                          .CLR( reset_local_FF ),
                            .Q( stop           ) );
      end // if
      if ( DEFAULT_VALUE == 1 ) begin : STOP_LATCH_1
         DFN1C1 stop_latch( .D( 1'b1           ),
                          .CLK( pulse_in       ),    // rising-edge triggered - end of
pulse
                          .CLR( reset_local_FF ),
                            .Q( stop           ) );
      end // if
   endgenerate

   // Mux to control ring-oscillator - chain #1
   MX2  chain1_mux(  .A( start                ),    // delay-line operation
                     .B( chain_1_out          ),    // ring-oscillator operation
                     .S( chain_1_loop_en      ),    // selection
                     .Y( vernier_chain_1[0]   ) );  // Start Vern Chain #1


   // Mux to control ring-oscillator - chain #2
   MX2  chain2_mux(  .A( stop               ),      // delay-line operation
                     .B( chain_2_out        ),      // ring-oscillator operation
                     .S( chain_2_loop_en    ),      // selection
                     .Y( chain_2_start      ) );    // start of vernier delay chain

   // Generate tap delay line for chain #1 : Using AO1 gate
   genvar i;
   generate
      for( i=0 ; i < N_STAGES   ; i=i+1 ) begin : chain_1  // this chain is slower

         AO1 chain_1(    .A( vernier_chain_1[i]   ),         //
                         .B( vernier_chain_1[i]   ),
                         .C( vernier_chain_1[i]   ),
                         .Y( vernier_chain_1[i+1] ) );
      end
   endgenerate

   // Extra gate delay on chain 1 to compenstate for edge detect logic delay
   BUFD chain2_extra( .A( chain_2_start ),
                      .Y( vernier_chain_2[0] ) );

   // Generate tap delay line for chain #2
   genvar j;
   generate
      for( j=0 ; j < N_STAGES  ; j = j + 1 ) begin : chain_2 // this chain is faster
         BUFD chain_2( .A( vernier_chain_2[j]   ),
                       .Y( vernier_chain_2[j+1] ) );
      end
   endgenerate

   // Generate the flops for capturing the vernier line
   genvar k;
   generate
      for( k=0 ; k < N_STAGES  ; k = k + 1 ) begin : flops
         DLI1 FF( .G( vernier_chain_1[ k+1 ] ),  // chain 1 slower (clocks the flops)
                  .D( vernier_chain_2[ k+1 ] ),  // chain 2 faster (data is racing to
catch up with the clock)
                .QN( data_out[k+4]        ) );
      end
   endgenerate

   // Generate bits #0 and #1 of data_out
   assign data_out[0] = ~start;
   assign data_out[1] = stop;

   // Pad remaining bits of the low nibble in data_out
   assign data_out[OUT_REG_WIDTH-1:N_STAGES+4] = 0;
   assign data_out[3:2] = 0;
```

```
   // Place inverters at the output of each chain - for use as ring-oscillator
   INVD chain1_inv(   .A( vernier_chain_1[ N_STAGES ] ),
                      .Y( chain_1_out                 ) );

   INVD chain2_inv(   .A( vernier_chain_2[ N_STAGES ] ),
                      .Y( chain_2_out                 ) );

   wire int_out_next;

   // Generate the interrupt indicate as the logical OR of start or stop
   OR2 U_OR( .A( data_out[0] ),
             .B( data_out[1] ),
             .Y( int_out_next ) );

   // Or tree for interrupt output
   always @( posedge clk ) begin
     int_out <= int_out_next;
   end

endmodule // VernierSETMonitor
```

Appendix B - HDL Source Code for Complex Circuits

B.1 – 8x8 Multiplier

```
module mult8( A, B, Y );

   input   [7:0] A;
   input   [7:0] B;
  output  [15:0] Y;

   assign Y = A * B;

endmodule // mult8
```

B.2 – 16-bit Adder

```
module add16( A, B, CIN, Y, COUT );

   input   [15:0] A;
   input   [15:0] B;
   input          CIN;
  output  [15:0]  Y;
  output          COUT;

  wire [16:0] SUM;

   assign SUM = { 15'b0, CIN } + A + B;
   assign Y = SUM[15:0];
   assign COUT = SUM[16];

endmodule // add16
```

B.3 – 32 Bit Priority Encoder

```
module PriEnc32( A, Y, NONE );

   input   [31:0] A;
  output   [4:0]  Y;
  output          NONE;

     wire [3:0] TOP_HALF, BOTTOM_HALF;

     assign Y[4] = ( A[15:0] == 16'd0 );

     assign TOP_HALF = A[31] ? 4'd15 :
                       A[30] ? 4'd14 :
                       A[29] ? 4'd13 :
                       A[28] ? 4'd12 :
                       A[27] ? 4'd11 :
                       A[26] ? 4'd10 :
                       A[25] ? 4'd9 :
                       A[24] ? 4'd8 :
                       A[23] ? 4'd7 :
                       A[22] ? 4'd6 :
                       A[21] ? 4'd5 :
                       A[20] ? 4'd4 :
                       A[19] ? 4'd3 :
                       A[18] ? 4'd2 :
                       A[17] ? 4'd1 :
                       A[16] ? 4'd0 : 4'bxxxx;

      assign BOTTOM_HALF = A[15] ? 4'd15 :
                           A[14] ? 4'd14 :
                           A[13] ? 4'd13 :
                           A[12] ? 4'd12 :
                           A[11] ? 4'd11 :
                           A[10] ? 4'd10 :
                            A[9] ? 4'd9 :
                            A[8] ? 4'd8 :
                            A[7] ? 4'd7 :
                            A[6] ? 4'd6 :
                            A[5] ? 4'd5 :
                            A[4] ? 4'd4 :
```

```
                            A[3] ? 4'd3 :
                            A[2] ? 4'd2 :
                            A[1] ? 4'd1 :
                            A[0] ? 4'd0 : 4'bxxxx;

      assign Y[3:0] = Y[4] ? TOP_HALF : BOTTOM_HALF ;

   assign NONE = Y[4] & ( A[31:16] == 16'd0 );

endmodule // PriEnc32
```

B.4 – State Machine

```
module Example_SMLogic(

        reset,            // Synchronous reset

        // State (old and new)
        current_state,
        next_state,

        // SM Inputs
        sm_input,

        // SM Outputs
        sm_output

   );

   parameter  SM_WIDTH  = 8;
   parameter  N_INPUTS  = 16;
   parameter  N_OUTPUTS = 4;

   input                    reset;
   input  [SM_WIDTH-1:0]    current_state;
   output [SM_WIDTH-1:0]    next_state;
   input  [N_INPUTS-1:0]    sm_input;
   output [N_OUTPUTS-1:0]   sm_output;

   reg    [SM_WIDTH-1:0] next_state;

   parameter  STATE0_STATE = 8'b00000001;  parameter STATE0_BIT_POSN = 0;
   parameter  STATE1_STATE = 8'b00000010;  parameter STATE1_BIT_POSN = 1;
   parameter  STATE2_STATE = 8'b00000100;  parameter STATE2_BIT_POSN = 2;
   parameter  STATE3_STATE = 8'b00001000;  parameter STATE3_BIT_POSN = 3;
   parameter  STATE4_STATE = 8'b00010000;  parameter STATE4_BIT_POSN = 4;
   parameter  STATE5_STATE = 8'b00100000;  parameter STATE5_BIT_POSN = 5;
   parameter  STATE6_STATE = 8'b01000000;  parameter STATE6_BIT_POSN = 6;
   parameter  STATE7_STATE = 8'b10000000;  parameter STATE7_BIT_POSN = 7;

   always @( * ) begin
      next_state = current_state;

      if (reset)
         next_state = STATE0_STATE;
      else begin
         case (1'b1)
            // State 0
            current_state[STATE0_BIT_POSN] : begin
               if (sm_input[0] && sm_input[1])
                  next_state = STATE1_STATE;
               else if (sm_input[2] || sm_input[3])
                  next_state = STATE2_STATE;
            end
            // State 1
            current_state[STATE1_BIT_POSN] : begin
               if (sm_input[2] && sm_input[3])
                  next_state = STATE2_STATE;
               else if (sm_input[2] || sm_input[3])
                  next_state = STATE3_STATE;
            end
            // State 2
            current_state[STATE2_BIT_POSN] : begin
               if (sm_input[4] && sm_input[3])
                  next_state = STATE3_STATE;
            end
            // State 3
            current_state[STATE3_BIT_POSN] : begin
```

```
               if (sm_input[5] && sm_input[3] && sm_input[6])
                  next_state = STATE0_STATE;
               else if (sm_input[2] || sm_input[3])
                  next_state = STATE1_STATE;
            end

            // State 4
            current_state[STATE4_BIT_POSN] : begin
                  next_state = STATE5_STATE;
            end
            // State 5
            current_state[STATE5_BIT_POSN] : begin
               if ( (sm_input[7] && sm_input[8]) || (sm_input[9]&&sm_input[10]) )
                  next_state = STATE6_STATE;
               else if (sm_input[2] || sm_input[3])
                  next_state = STATE2_STATE;
            end
            // State 6
            current_state[STATE6_BIT_POSN] : begin
               if ( sm_input[11] || sm_input[12] || sm_input[13] )
                  next_state = STATE7_STATE;
               else if (sm_input[5] || sm_input[7])
                  next_state = STATE1_STATE;
            end
            // State 7
            current_state[STATE7_BIT_POSN] : begin
               if ( sm_input[14] || sm_input[15] )
                  next_state = STATE0_STATE;
            end

         endcase
      end
   end

   assign sm_output[0] = current_state[STATE0_BIT_POSN] || current_state[STATE1_BIT_POSN]
;

   assign sm_output[1] = &sm_input[3:0] & current_state[STATE0_BIT_POSN];

   assign sm_output[2] = ( current_state[STATE5_BIT_POSN] ||
current_state[STATE3_BIT_POSN] ) && ( sm_input[3:0] > 4'd5 );

   assign sm_output[3] = ( current_state[STATE3_BIT_POSN] && ( sm_input[2] || sm_input[1]
) ) ||
                         ( current_state[STATE1_BIT_POSN] && ( sm_input[0] ^ sm_input[3]
) ) ;

endmodule // Example_SMLogic
```

B.5 – 16-bit Magnitude Comparator

```
module MagCompare16( A, B, EQ, GT );
   input [15:0] A;
   input [15:0] B;
   output       EQ;
   output       GT;

   assign EQ = ( A == B );
   assign GT = ( A  > B );

endmodule // MagCompare16
```

B.6 – 4-Bit 8:1 Mux

```
module Mux8_4bit( in, out, sel );
   input  [31:0] in;
   output  [3:0] out;
   input   [2:0] sel;

   reg [3:0] out;

   always @( * ) begin
      case ( sel )
           0 : out = in[3:0];
           1 : out = in[7:4];
           2 : out = in[11:8];
```

```
            3 : out = in[15:12];
            4 : out = in[19:16];
            5 : out = in[23:20];
            6 : out = in[27:24];
            7 : out = in[31:28];
      default : out = 3'bxxx;
      endcase
  end // always

endmodule // Mux8_4bit
```

B.7 16-Bit Hamming ECC Encoder

```
module ecc_SEC16_encoder ( DIN , ECC );
   // Port Declarations
   input [15:0] DIN;
   output [4:0] ECC;

   // Signal Declarations
   // Parity Bit Calculation
   assign ECC[0] = DIN[0] ^ DIN[1] ^ DIN[3] ^ DIN[4] ^ DIN[6] ^ DIN[8] ^ DIN[10] ^
DIN[11] ^ DIN[13] ^ DIN[15];
   assign ECC[1] = DIN[0] ^ DIN[2] ^ DIN[3] ^ DIN[5] ^ DIN[6] ^ DIN[9] ^ DIN[10] ^
DIN[12] ^ DIN[13];
   assign ECC[2] = DIN[1] ^ DIN[2] ^ DIN[3] ^ DIN[7] ^ DIN[8] ^ DIN[9] ^ DIN[10] ^
DIN[14] ^ DIN[15];
   assign ECC[3] = DIN[4] ^ DIN[5] ^ DIN[6] ^ DIN[7] ^ DIN[8] ^ DIN[9] ^ DIN[10];
   assign ECC[4] = DIN[11] ^ DIN[12] ^ DIN[13] ^ DIN[14] ^ DIN[15];
endmodule // ecc_SEC16_encoder
```

B.8 16-Bit Hamming ECC Decoder

```
module ecc_SEC16_decoder ( DIN , ECC , DOUT );

   // Port Declarations
   input [15:0] DIN /* synthesis syn_keep=1 */ ;
   input [4:0] ECC /* synthesis syn_keep=1 */ ;
   output [15:0] DOUT /* synthesis syn_keep=1 */ ;

   // Signal Declarations
   wire  [4:0] SYN /* synthesis syn_keep=1 */ ;
   wire  [15:0] CORR /* synthesis syn_keep=1 */ ;

   // Syndrome Bit Calculation
   assign SYN[0] = ECC[0] ^ DIN[0] ^ DIN[1] ^ DIN[3] ^ DIN[4] ^ DIN[6] ^ DIN[8] ^ DIN[10]
^ DIN[11] ^ DIN[13] ^ DIN[15];
   assign SYN[1] = ECC[1] ^ DIN[0] ^ DIN[2] ^ DIN[3] ^ DIN[5] ^ DIN[6] ^ DIN[9] ^ DIN[10]
^ DIN[12] ^ DIN[13];
   assign SYN[2] = ECC[2] ^ DIN[1] ^ DIN[2] ^ DIN[3] ^ DIN[7] ^ DIN[8] ^ DIN[9] ^ DIN[10]
^ DIN[14] ^ DIN[15];
   assign SYN[3] = ECC[3] ^ DIN[4] ^ DIN[5] ^ DIN[6] ^ DIN[7] ^ DIN[8] ^ DIN[9] ^
DIN[10];
   assign SYN[4] = ECC[4] ^ DIN[11] ^ DIN[12] ^ DIN[13] ^ DIN[14] ^ DIN[15];

   // Correction Bit Calculation
   assign CORR[0] = SYN[0] && SYN[1] && ( ! SYN[2] ) && ( ! SYN[3] ) && ( ! SYN[4] );
   assign CORR[1] = SYN[0] && ( ! SYN[1] ) && SYN[2] && ( ! SYN[3] ) && ( ! SYN[4] );
   assign CORR[2] = ( ! SYN[0] ) && SYN[1] && SYN[2] && ( ! SYN[3] ) && ( ! SYN[4] );
   assign CORR[3] = SYN[0] && SYN[1] && SYN[2] && ( ! SYN[3] ) && ( ! SYN[4] );
   assign CORR[4] = SYN[0] && ( ! SYN[1] ) && ( ! SYN[2] ) && SYN[3] && ( ! SYN[4] );
   assign CORR[5] = ( ! SYN[0] ) && SYN[1] && ( ! SYN[2] ) && SYN[3] && ( ! SYN[4] );
   assign CORR[6] = SYN[0] && SYN[1] && ( ! SYN[2] ) && SYN[3] && ( ! SYN[4] );
   assign CORR[7] = ( ! SYN[0] ) && ( ! SYN[1] ) && SYN[2] && SYN[3] && ( ! SYN[4] );
   assign CORR[8] = SYN[0] && ( ! SYN[1] ) && SYN[2] && SYN[3] && ( ! SYN[4] );
   assign CORR[9] = ( ! SYN[0] ) && SYN[1] && SYN[2] && SYN[3] && ( ! SYN[4] );
   assign CORR[10] = SYN[0] && SYN[1] && SYN[2] && SYN[3] && ( ! SYN[4] );
   assign CORR[11] = SYN[0] && ( ! SYN[1] ) && ( ! SYN[2] ) && ( ! SYN[3] ) && SYN[4];
   assign CORR[12] = ( ! SYN[0] ) && SYN[1] && ( ! SYN[2] ) && ( ! SYN[3] ) && SYN[4];
   assign CORR[13] = SYN[0] && SYN[1] && ( ! SYN[2] ) && ( ! SYN[3] ) && SYN[4];
   assign CORR[14] = ( ! SYN[0] ) && ( ! SYN[1] ) && SYN[2] && ( ! SYN[3] ) && SYN[4];
   assign CORR[15] = SYN[0] && ( ! SYN[1] ) && SYN[2] && ( ! SYN[3] ) && SYN[4];

   // Data correction
   assign DOUT[0] = DIN[0] ^ CORR[0];
   assign DOUT[1] = DIN[1] ^ CORR[1];
   assign DOUT[2] = DIN[2] ^ CORR[2];
   assign DOUT[3] = DIN[3] ^ CORR[3];
```

```
   assign DOUT[4] = DIN[4] ^ CORR[4];
   assign DOUT[5] = DIN[5] ^ CORR[5];
   assign DOUT[6] = DIN[6] ^ CORR[6];
   assign DOUT[7] = DIN[7] ^ CORR[7];
   assign DOUT[8] = DIN[8] ^ CORR[8];
   assign DOUT[9] = DIN[9] ^ CORR[9];
   assign DOUT[10] = DIN[10] ^ CORR[10];
   assign DOUT[11] = DIN[11] ^ CORR[11];
   assign DOUT[12] = DIN[12] ^ CORR[12];
   assign DOUT[13] = DIN[13] ^ CORR[13];
   assign DOUT[14] = DIN[14] ^ CORR[14];
   assign DOUT[15] = DIN[15] ^ CORR[15];

endmodule // ecc_SEC16_decoder
```