

New insights into the impact of SEUs in FPGA CRAMs

Sheng Wang¹, Adrian Evans², Shi-Jie Wen³,
Rick Wong³, and GengSheng Chen^{1a)}

¹ ASIC and System State Key Lab, Fudan University,

ME BLD, NO.825 Zhangheng RD, Pudong District, Shanghai 201203, China

² IROC Technologies,

World Trade Center, PO BOX 1510, Grenoble, Isere 38025, France

³ Cisco Systems,

Corporate Headquarters, 70 West Tasman Dr., San Jose, CA 95134, United States

a) gschen@fudan.edu.cn

Abstract: This paper presents a detailed study of the impact of SEUs in the configuration RAM (CRAM) of SRAM based FPGAs. Since modern SRAM based FPGAs support scrubbing of the CRAM, a new, intermittent CRAM SEU fault model is presented. This fault model is implemented both in simulation and on an emulation platform for an embedded processor design. The criticality of CRAM bits is studied based on their logic function, the duration of the SEU, and the workload running on the processor. These results provide new insight into the overall effectiveness of CRAM scrubbing mechanisms.

Keywords: FPGA, SEU, fault-injection, reliability

Classification: Integrated circuits

References

- [1] Xilinx: “LogiCore IP Soft Error Mitigation Controller v4.1 – Product Guide.” PG036 (2014).
- [2] T. Imagawa, M. Hiromoto, O. Hiroyuki and T. Sato: IEICE Trans. Fundamentals **E93-A** (2010) 2524. DOI:10.1587/transfun.E93.A.2524
- [3] J. Napoles, H. Guzman-Miranda, M. Aguirre, J. N. Tombs, J. M. Mogollon, R. Palomo and A. P. Vega-Leal: 2008 4th Southern Conference on Programmable Logic (2008) 213.
- [4] L. Sterpone and M. Violante: IEEE Trans. Nucl. Sci. **52** (2005) 2217. DOI:10.1109/TNS.2005.860745
- [5] Y. Ichinomiya, T. Kimura, M. Amagasaki, M. Iida and T. Sueyoshi: IEICE Trans. Fundamentals **E95-A** (2012) 2347. DOI:10.1587/transfun.E95.A.2347
- [6] H. Cho, S. Mirkhani, C.-Y. Cher, J. A. Abraham and S. Mitra: Design Automation Conference (DAC) (2013).
- [7] Xilinx. “Device Reliability Report.” UG116 (2014).
- [8] A. Manuzzato, S. Gerardin, A. Paccagnella, L. Sterpone and M. Violante: IEEE Trans. Nucl. Sci. **55** (2008) 1968. DOI:10.1109/TNS.2008.2000850
- [9] M. Alderighi, F. Casini, S. D’Angelo, M. Mancini, D. M. Codinachs, S. Pastore, G. Sorrenti, L. Sterpone, R. Weigand and M. Violante: 2008 European Conference on RADECS (2008) 157. DOI:10.1109/RADECS.2008.5782703
- [10] M. Alderighi, F. Casini, S. D’Angelo, M. Mancini, D. M. Codinachs, S.

- Pastore, C. Poivey, G. R. Sechi, G. Sorrenti and R. Weigand: IEEE Trans. Nucl. Sci. **57** (2010) 2129. DOI:10.1109/TNS.2010.2043855
- [11] M. Violante, L. Sterpone, M. Ceschia, D. Bortolato, P. Bernardi, M. S. Reorda and A. Paccagnella: IEEE Trans. Nucl. Sci. **51** (2004) 3354. DOI:10.1109/TNS.2004.839516
- [12] L. Sterpone, M. Violante and S. Rezgui: IEEE Trans. Nucl. Sci. **53** (2006) 2054. DOI:10.1109/TNS.2006.880937
- [13] O. Héron, T. Arnaout and H. Wunderlich: Field Programmable Logic and Applications, 2005. International Conference (2005) 403. DOI:10.1109/FPL.2005.1515755
- [14] University of Michigan Mibench suite: <http://www.eecs.umich.edu/mibench/>.

1 Introduction

SRAM-based FPGA devices are widely used in industry and academic research in various environments. With technology scaling, there is an increasing interest in using SRAM-based FPGAs in safety-critical, high-reliability and space applications, where the impact of radiation-induced upsets is a concern. It is important to make the distinction between upsets that affect elements in the data-path (flip-flops and block RAMs) and upsets that affect the Configuration RAM (CRAM), which defines the actual logic function of the FPGA. The latter are of greater concern due to the enormous number of CRAM bits.

Vendors are well aware of this problem and modern FPGA families have support for background scrubbing mechanisms which traverse the CRAM and identify single (or even double, adjacent) bit errors [1]. However, there is still a window of time between when the error occurs and when it is detected, during which the logic function may not be correct. Even though the scrubbing process can detect and correct the error, the implemented logic function was incorrect during this window of time, which may result in corrupted system state and incorrect results.

There is limited research on the relationship between the duration of a CRAM fault and its impact on the overall system. In this paper, we present an in-depth study of the impact of CRAM bit upsets. The main contribution is that we evaluate the impact of the *duration* that the fault is present and perform a *fine-grained* analysis of the sensitivity of different types of CRAM bits. Results are generated and analyzed with simulation techniques and emulated hardware fault injection. These results are important for establishing acceptable scrubbing intervals and for understanding the most effective techniques for protecting the CRAM.

The remainder of the paper is organized as follows. In section 2, we review existing work on the analysis of SEUs in SRAM based FPGAs. In section 3 we describe the FDP FPGA, classify the types of CRAM bits and show how faults in different types of CRAM bits can be modeled in a simulator and can be injected to a design running on the FPGA. In section 4 we present the experimental results of a fault injection campaign performed on an 8051 micro-controller. This includes an analysis of the critical configuration bits, a study of the dependence of the fault duration on the probability of error and an analysis of the influence of the

workloads running on the processor. Conclusions and plans for future work are presented in section 5.

2 State of the art

Numerous platforms exist for performing simulation and emulation-based fault injection. One must differentiate those that inject faults on the data-path (flip-flops, gates) [2, 3] versus those that target upsets in the CRAM of FPGAs [1, 4, 5, 6].

Modern FPGAs have sophisticated mechanisms for detecting SEUs in the CRAM, including background scrubbing. The detection time for faults can exceed 50 msec even with the fastest clock frequency [1]. During this interval, the device may not operate correctly. Even though the fault is eventually corrected, the circuit's state may be corrupted and the circuit may produce incorrect results. In natural radiation environments, including space environments [7], the scrub interval is sufficiently short, to ensure that the probability that multiple upsets will accumulate is virtually zero. Despite this, we note that much research is focused on the accumulation of faults [8, 9], which is not realistic for devices which support scrubbing. Under accelerated radiation testing, where particles fluxes are hundreds of thousands of times higher than real radiative environments, multiple events may occur during the scrubbing interval. Thus the results of such testing do not accurately reflect the actual behavior in a natural environment, when the CRAM bits are scrubbed. This underscores the importance of simulation and emulation based fault-injection techniques when analyzing the circuit or system level impact of SEUs in the CRAM of SRAM-based FPGAs.

In [5], partial reconfiguration is used to evaluate the effect of CRAM faults on a processor. The emphasis is on fault accumulation and a single workload is studied. Fault injection speed is limited by the cable setup time, which is over one second.

In [10] the Flipper system for Virtex-2 devices is described, arithmetic and signal processing circuits are studied and the results compared with radiation testing, however, scrubbing is not addressed. In [2], the authors consider selective TMR in FPGAs but the faults they consider are only on the data-path and not in the CRAM.

In [11] and [12], the authors analyze the effect of different types of CRAM bits, however, these studies are limited to combinatorial circuits and there is no analysis of the duration of the fault. In [13], the authors derive a mathematical model for determining the fraction of critical bits. They apply this model to the configuration bitstream for several complex designs in order to estimate the probability of failures. But the results are not validated with actual fault-injection tests, the mitigating effect of scrubbing is not considered and with a theoretical model it is not possible to evaluate the effect of the workloads running on the processor.

3 FPGA and slice

In this section, a fault emulation and simulation method for faults on the CRAM bits of SRAM-based FPGA is introduced.

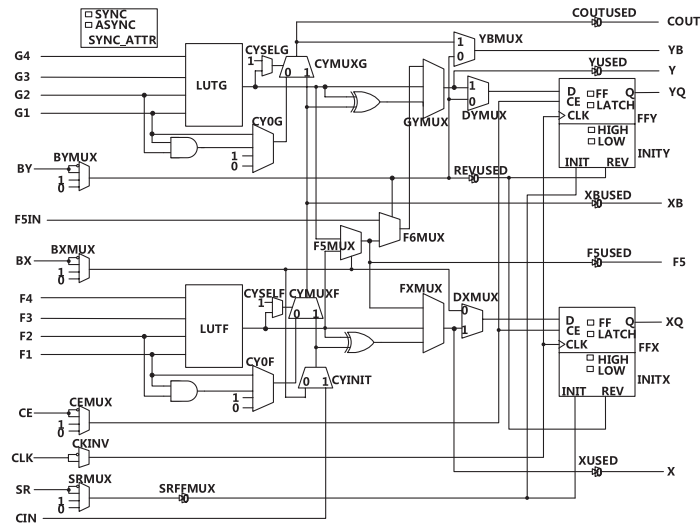


Fig. 1. Basic logic slice

3.1 CRAM bits in SRAM-based FPGAs

The FDP FPGA, used in this study, is an academic FPGA being designed for research purposes. It has 32×48 CLBs with each CLB having two identical logic slices. Each slice has two 4-inputs LUTs and two flip-flops. The circuit diagram for the basic logic slice is shown in Fig. 1.

The FDP has a total of 1,670,864 CRAM bits. The focus of this study is on the impact of SEUs on the CRAM bits, specifically on those CRAM bits that control the logic function, thus those bits controlling the PLL and IOBs are not considered. For a typical design implemented on the FDP, only a fraction of the total CRAM bits are used. Those CRAM bits associated with the resources are not essential to the design. Table I shows the number of CRAM bits that are actually used by several application circuits on the FDP.

Table I. Essential bits

Circuit	Occupied Slices	Essential Bits	Total CRAM Bits	Percentage
8051	3025	410710	1670864	24.6%
32-bit multiplier	544	66529	1670864	3.98%
32-bit adder	16	3994	1670864	0.24%

3.2 SEUs and fault emulation

During the scrubbing window, logic faults caused by a SEU may cause bad state to be propagated in the circuit. This transitory upset can be modeled and emulated by injecting a fault on a specific CRAM bit, at a specific time with a certain duration. Thus, each fault is modeled as a tuple consisting of: $(location, t_{fault}, t_{duration})$. The location specifies which CRAM bit is upset (inverted). The value of t_{fault} specifies at what point in time the fault is injected. The value of $t_{duration}$ specifies the duration of time before the fault is removed. It is important to note that in the experiments of this work, faults are *only* injected on those CRAM bits that are actually used by the design (essential bits), specified in a *MarkFile* that is generated by custom tools.

The FDP FPGA supports frame-based Dynamic Partial Reconfiguration (DPR), with a basic frame unit of 20-words. With the mechanism of frame-based PR, a given fault can be dynamically injected to a specific CRAM bit anytime without any interruption to the application circuits on the FDP.

3.3 Fault generation

Before faults are injected in a design, custom tools are used to generate the faults and insert them to the downloadable bitstreams. The tool flow for the methodology is shown in Fig. 2. After synthesis, custom tools are used to generate the FPGA netlist, which contains physical design information. The netlist conversion tool (NetConvert) converts the netlist to its Verilog-HDL version for gate-level simulation. The bitstream generation tool (BitGen) receives the netlist and generates the fault-free bitstream.

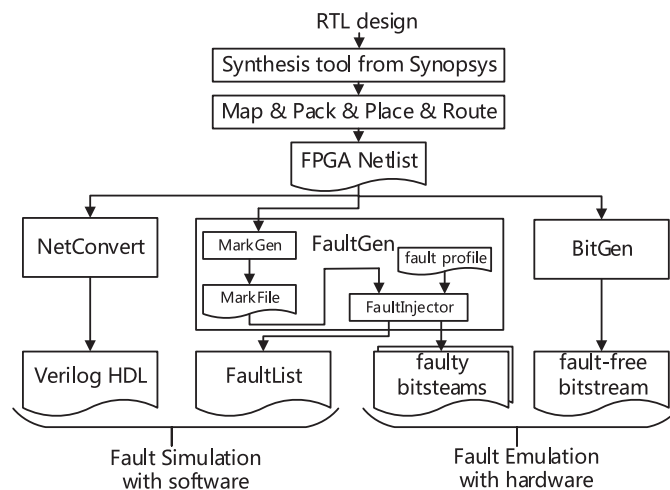


Fig. 2. Tool flow for fault injection

FaultGen is a tool developed to generate the FaultList and corresponding faulty bitstreams. FaultGen has two components, MarkGen and FaultInjector. MarkGen generates the *MarkFile* which keeps the location information of all essential bits used by the application design. The fault profile specifies the number of faults to be injected to each type of programmable logic resources (i.e. CRAM bit for DYMUX). Using the information from the *MarkFile* and fault profile, FaultInjector generates a set of faulty bitstreams. Therefore, a fault can be injected on any position of the design on demand.

In Fig. 2, the Verilog-HDL netlist is used to simulate the faults, while the faulty and fault-free bitstreams are used for fault emulation. Using scripts, the fault injection campaigns can be run automatically.

3.4 Simulation-based analysis of CRAM SEU effects

Correctly modeling the effect of a *transitory* upset to the CRAM is non-trivial as the original and modified circuit must co-exist and the transition between them must occur in a realistic fashion. In our methodology, additional logic is automatically inserted into the netlist to model the fault. To the authors' knowledge, this is

the first work which describes how transitory faults in CRAM bits can be correctly modeled in simulation which is different from the other systems that model permanent CRAM faults and accumulated CRAM faults.

Consider the example shown in Fig. 3, where a fault is injected to a CRAM bit controlling a LUT. The logic for the faulty LUT configuration and an additional MUX are inserted into the circuit. The new input, *Fault_sel*, is used to select between the two versions of the logic. The process of reading the original design and inserting MUXes and faulty components to inject a specified set of faults has been fully automated using Perl scripts. In the cases where the fault affects a sequential element (for example, a CRAM flip which changes a flip flop into a latch), both versions of the sequential element are simulated. This ensures that the faulty version has the correct state at the time when the fault is injected. All simulations in this work are performed using ModelSim.

Digital simulators cannot accurately model designs with undriven inputs. However, as the result of a CRAM bit flip, an output may be disconnected, creating a floating net, thus making the design non-deterministic. In a real circuit, the undriven net may float high or low depending on the parasitic capacitances specific to the given die. In the simulation platform, both the cases of the floating wire being at 0 and 1 are evaluated and the more pessimistic of the two outcomes is selected.

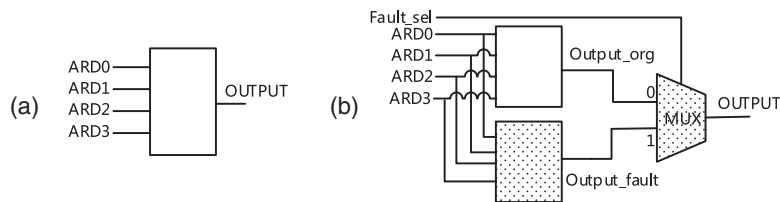


Fig. 3. (a) Original, (b) Modified circuit

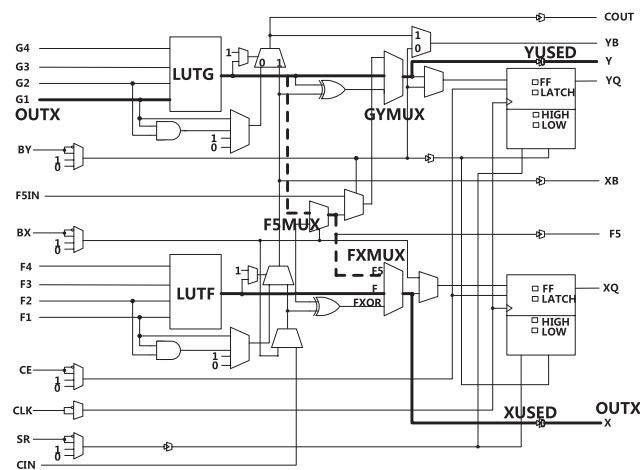


Fig. 4. Unintended combinational loop

Another interesting type of fault occurs when the upset of a CRAM bit causes a combinational loop, unintentionally creating a ring-oscillator. Consider the original logic function shown in Fig. 4. The original circuit is shown with solid dark lines. LUTF's output "OUTX" drives one of LUTG's inputs through FXMUX and

interconnection wires. If an upset causes the FXMUX to select “F5” rather than “F”, as shown with the dashed line, a combinatorial loop through LUTG is produced. In hardware, this case results in increased power consumption. In the event based simulator, it causes a timeout when the iteration limit is reached.

3.5 Hardware emulation platform

Fig. 5 shows the block diagram of the hardware platform for performing CRAM fault emulation and injection. The hardware platform consists of four parts: the FDP device where the application circuit (CUT, circuit under test) is, a controlling card (an Altera DE4 card) acting as the testbench, a network interface (W7200), and a host computer (PC). The FIC (Fault Injection Controller) is a software program on the PC. It combines the faulty bitstreams, inserts command and sync segments, and generates a downloadable bitstream for the FDP device. The FIA (Fault Injection Agent) is a circuit on the DE4. Synchronized by the FIC, the FIA feeds stimuli to the CUT and records the collected response in DP_RAM memory. The experimental data are read and forwarded to the PC by the W7200. Software on the PC saves the data for analysis. The FIC synchronizes the overall process.

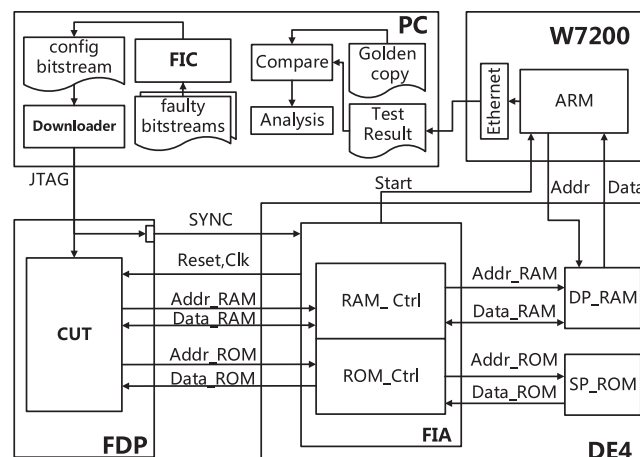


Fig. 5. Hardware platform for fault injection

In this work, an 8051 micro-controller (*processor*) is used as the CUT mapped to the FDP FPGA and requiring 410,710 CRAM bits. The *processor*, clocked at a frequency of 0.5¹ MHz, has the SP_ROM (single port ROM) on the DE4 as its external ROM and has the DP_RAM (dual-port RAM) on the DE4 as its external RAM. Three programs (ISQRT, LONG2BIN, SORT) from the MiBench suite [14], stored in the SP_ROM, served as the *processor*’s workloads. The programs were modified to signal their completion and results checking was performed by reading back the RAM contents and comparing it with known good data.

The result of each fault injection test was classified into one of four categories, based on a subset of the categories in [6]. If the results are correct at the normal completion time of the program, the fault injection is labeled as ‘On Time Correct’. If the program indicates it has finished, but the results are wrong, the fault is labeled

¹The frequency was limited by the experimental FDP; whereas this frequency does not invalidate the results. The finding about the scrubbing interval can be scaled to designs at higher frequencies.

as ‘*Wrong Result*’. If the program fails to finish (after $>1.2\times$ the normal execution time), it is labeled as ‘*Did not Complete*’. If the program finishes late ($\leq 1.2\times$ normal time) but the results are correct, it is labeled as ‘*Late but Correct*’. If it finishes late ($> 1.2\times$ normal time) with wrong results, the fault injection is also labeled as ‘*Wrong Result*’. Unlike the LEON3 processor used in [6], fault detection through traps or exceptions do not exist in the 8051 micro-controller, thus the corresponding category in [6] is not considered in this work.

3.6 Hardware-based CRAM SEU effect analysis

Using the 8051 micro-controller as the CUT, Fig. 6 shows the process flow diagram for the SEU effect analysis. At the FIC side, the config file contains the full configuration bitstream of the 8051 and the bitstream for N (1 to 8000) iterations of fault injection. The config file is continuously downloaded to the FDP until the entire test has finished. At the CUT and the FIA side, the FIA, synchronizes to the FIC with the SYNC signal, controls the CUT and the W7200 to finish the emulation process.

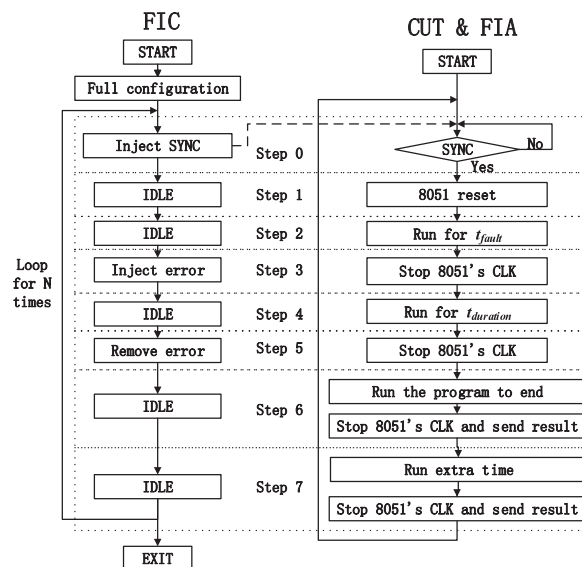


Fig. 6. Flow diagram for hardware fault injection

First, the full configuration of the 8051 is downloaded to the FDP. Then, the FIC sends the SYNC signal to the FIA (step 0). The *processor* on FDP is reset (step 1). The *processor* advances for a time equal to t_{fault} to reach the point when the fault should be injected (step 2). Then, the clock is stopped, and the configuration with the inverted CRAM bit is downloaded using the DPR (step 3). The clock is then resumed and the *processor* is allowed to run for $t_{duration}$, in order to model the time during which the fault is present (step 4). The clock is again stopped and the original configuration is downloaded again to the FPGA (step 5). The *processor*'s clock is again resumed until the instant in time corresponding to the normal completion time. At this point, the contents of the memory (recorded results) are read back and sent to the PC to be verified (step 6). The clock is then resumed and the *processor* is allowed to run for some additional time, after which

the recorded results are again dumped (step 7). After step 7, the FIA goes back to step 0 waiting for another SYNC signal from the FIC. The recorded results are dumped twice in order to determine whether the program will produce current results, given extra time.

3.7 Validation of fault emulation and simulation

A combination of the hardware fault emulation platform and the software simulation platform are used in the experiments for fault injection and analysis. The hardware emulation platform is used to quickly evaluate the impact of SEUs, while the simulation platform is used to perform fine-grained analysis, which is effective for both fault coverage and in-depth investigation.

For the purpose of validating the emulation platform, 2100 CRAM bits were selected randomly from the 410,710 CRAM bits. For each of these faults (*location*, t_{fault} , t_{duration}), it was ensured that the error produced in simulation was identical to that produced on the emulation platform. Note that non-deterministic faults, such as “undriven inputs” were excluded from this comparison.

4 Experimental results

We present the experimental results of a fault injection campaign performed on an 8051 micro-controller and in-depth study of the impact of CRAM faults. The results were obtained using both the hardware fault emulation platform and the software simulation platform introduced in section 3.

4.1 Criticality of CRAM bits

For a CRAM bit, if an upset results in a functional fault to the circuit, the bit is classified as “critical”. CRAM bits can be divided into three broad groups according to their usage: LUTs, interconnect (PIP) and MUXes. The MUXes can be further sub divided into 23 types (eg. DXMUX, DYMUX, CYINIT, FFX, etc.).

To obtain the criticality of each CRAM group, a total of 6,022 CRAM bits (*location*) were randomly selected. The LONG2BIN program was used as the workload. In this experiment, t_{fault} was selected randomly, while t_{duration} had a fixed value of 50 msec. The result of the criticality in percentage is given in Fig. 7.

From Fig. 7, it is observed that the upsets in the MUXes are more likely to result in incorrect results, thus the CRAM bits for MUXes are more critical than those for PIPs and LUTs. In [12], the CRAM bits for MUXes were also found to be more critical than those for LUTs but the CRAM bits for PIP were the most critical.

Fig. 8 shows a more detailed analysis of the criticality of different CRAM bits by function. Overall, the DYMUX, which controls the feed-in of the flip flop, as shown in Fig. 1, is the most critical CRAM bit. We note that the DYMUX appears more sensitive than the DXMUX, despite the fact they perform essentially the same function. As a result of the tool’s packing strategy, on average, each FFY drives 4.9 ports whereas each FFX drives only 3.0 ports. The higher fanout of the DYMUX explains its increased sensitivity compared to the DXMUX. The CYINIT and CYSELF, which control the carry-in (CIN) and carry-out (COUT) were also found to be very sensitive.

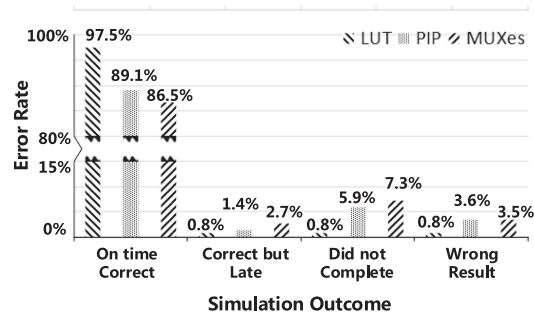


Fig. 7. CRAM criticality by category

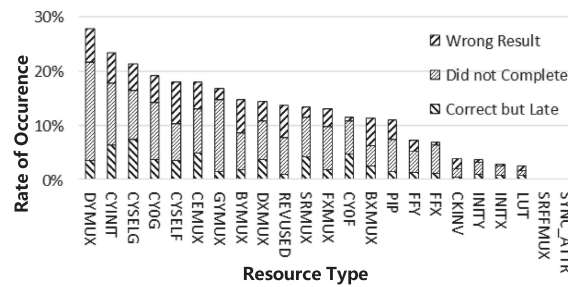


Fig. 8. Error rate versus CRAM bit function

4.2 Impact of fault duration

Understanding the relationship between fault duration and the probability of error is important for establishing optimal scrubbing intervals. Higher scrubbing rates require additional hardware resources and result in increased power consumption. In this experiment, three programs (ISQRT, LONG2BIN, SORT) were used as the workloads. 6,000 CRAM bits (*location*) from all the essential bits were randomly selected. Five different values of $t_{duration}$ (20 μ sec, 200 μ sec, 2 msec, 20 msec, 50 msec) were used. Thus, 90000 fault injections were performed on the 6000 CRAM bits for three different workloads, where t_{fault} was selected randomly.

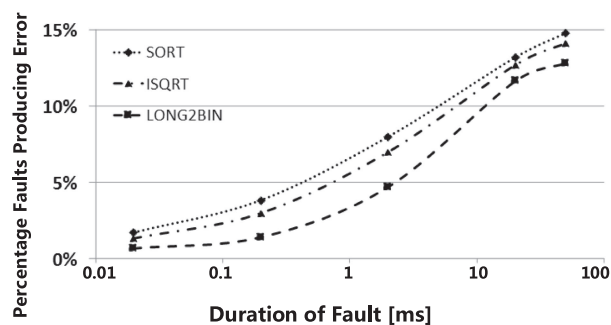


Fig. 9. Error rate versus fault duration

In Fig. 9, the error rate is graphed as a function of the fault duration. As expected, as the duration of the fault increases, the rate of errors also increases.

From Fig. 9, it is clear that the impact of the fault increases quickly with its duration. After approximately 2 msec (1000 clock cycles), the probability that a

CRAM upset causes an error starts to increase significantly. For a processor running at a more typical frequency of 100 MHz, 1000 clock cycles corresponds to 10 μ sec, which is shorter than typical scrubbing intervals. This suggests that a scrubbing with an interval of 50 msec on commercial FPGAs may not be effective at preventing incorrect results.

4.3 Impact of workload

In Fig. 9 it can be seen that the sensitivity of the SORT program is always the highest regardless of the fault duration. Thus, the relative sensitivity of the applications appears independent of the fault duration.

Hardware designers often have limited knowledge of the application software, thus, it is important to understand whether the CRAM bits that are critical for one application are also critical for others. Using the three programs (ISQRT, LONG2BIN, SORT), experiments were performed where a total of 30,000 faults were injected on 1,000 randomly selected CRAM bits, with 10 faults (random value of t_{fault}) being injected to each CRAM bit and $t_{duration}$ fixed at 20 msec. The bits were then classified based on whether an error occurred in one, two or all three applications. The results are shown, as a Venn diagram, in Fig. 10.

From Fig. 10, we see that in total of 193 CRAM bits are found to be critical. Of these only about half were critical in any one application (from 103 to 107). So it is not sufficient to use a single application to identify all the critical bits. By using two programs, unique critical bits (147 to 164) were identified (76% to 85%). Thus, a minimum of two different applications are required to identify a reasonable set of critical CRAM bits.



Fig. 10. Venn diagram of critical CRAM bits

4.4 Performance evaluation of fault injection

With the hardware platform, using DPR, the effect of intermittent faults in the CRAM can be evaluated quickly. Since we allow the program to run 20% longer than normal to assess whether it will eventually produce the correct result, there is an intrinsic run time penalty. Table II summarizes the run time penalty when injecting faults for all three programs. The mean time per fault (MTPF), around 0.6 msec, in this work is significantly faster than [5]. The penalty on the MTPF is under 24% of the normal run time, including the fixed 20% overhead.

Table II. Run time profiling

Workloads	Test case run time (msec)	Extra-run time (msec)	MTPF (msec)	Time Overhead for FI
ISQRT	450.3	90.1	556.7	23.6%
LONG2BIN	530.2	106.0	652.5	23.1%
SORT	432.0	86.4	534.7	23.8%

5 Conclusions and plans for future work

In this paper, we present a new platform that can perform fixed duration CRAM fault injections. To our knowledge, this is the only platform that models intermittent CRAM faults. This fault model is more realistic, since modern FPGAs contain a scrubbing engine which restores CRAM errors.

The hardware platform is validated through a fine-grained comparison with simulation. The latter was invaluable as it enabled special cases, such as the combinatorial loop situation illustrated in Fig. 4, to be understood. Developing a platform for evaluating intermittent CRAM faults is significantly more complex than modeling permanent faults. The simulation platform must switch between the correct and incorrect design while preserving the state between the two designs.

The focus of this paper is the analysis of the criticality of faults and the results are helpful when designing lower cost error mitigation techniques. Knowledge of which CRAM bits are most critical makes it possible to selectively harden a subset of the bits using circuit techniques. The benefit of hardening a DYMUX CRAM bit is much greater than that of hardening the INITX or INITY bits. Future work will present the results showing the benefit of selective hardening of CRAM bits.

An understanding of the relationship between fault duration and the probability of error is extremely important. Our results suggest that for commercial designs working in the range of 100 MHz, a scrub interval on the order of 10 μ sec is required to significantly reduce the effect of faults. Using the sensitivity of different types of CRAM bits opens the possibility of implementing strategies, where the scrubbing rates are adapted based on the criticality of the CRAM resource.

To enhance the reliability of micro-controllers running on FPGAs, it is important to find the critical CRAM bits. An analysis of the impact of the workload to find critical CRAM bits was presented. A minimum of two different applications are required to identify a reasonable set of critical CRAM bits.

The mean time per fault (MTPF) in this work is about 0.6 msec, thus significantly faster than [5] (≈ 2 seconds). The run time penalty is only about 24% including a fixed 20% overhead, making it possible to evaluate a large number of faults.

Future work will be based both on larger, commercial FPGAs as well as on a next generation of the FDP. The benefit of selective CRAM protection will also be investigated.

Acknowledgments

The authors wish to acknowledge the financial support of Cisco Systems.