Study of the Effects of SET Induced Faults on Submicron Technologies

Alireza Rohani and Hans. G. Kerkhoff
Testable Design and Test of Integrated Systems Group
CTIT, University of Twente
Enschede, Netherlands
a.rohani@utwente.nl, H.G.Kerkhoff@utwente.nl

***Abstract*—the progression of shrinking technologies into processes below 100nm has increased the importance of transient faults in digital systems. Fault injection into the HDL model of the system, known as simulation-based fault injection, is being increasingly used in recent years in order to evaluate the behaviour of systems in the presence of transient faults. However, there are still several questions in conducting simulation-based fault injections. For instance, what is the importance of timing information of the netlist with regard to the accuracy of fault injection results? And how does the number of fault injection experiments relate to obtain a realistic behaviour of the processor under test. Finally, what is the dependence of fault injection results on the processor's workload? This paper aims to answer these questions, by studying the effects of transient faults on a post placed-and-routed Verilog netlist of a high performance reconfigurable processor in 90-nanometer UMC technology.**

Keywords-fault injection; fault simulations; Soft Errors; SET

I. INTRODUCTION

Shrinking the implementation of digital systems towards aggressive nanometer technologies has increased the importance of transient faults. Transient faults can arise from several sources like extraterrestrial cosmic rays (e.g. protons and neutrons), Electro Migration Interference (EMI), or internal sources such as alpha particles within the packaging materials [1]. If these high-energy sources strike a semiconductor, they progressively lose their energy whilst ionizing the medium. This process may alter data in combinational or sequential parts of a circuit. In the past, transient faults were of concern for those systems used in electronics-hostile environments such as outer space [2]. However, the characteristics of future-generation systems raise the susceptibility of digital systems even at sea level. Some of those characteristics include scaling of CMOS technology, decreasing operating voltage, increasing the complexity of digital systems and increasing the clock speed. Moreover, substantial use of wireless technologies like Wi-Fi and mobile phones has increased the hostility of the environment as a threat to nanometers digital systems [3]. As a conclusion, the problem of transient faults is forecast to be a severe problem for future digital systems even in conventional environments [3].

Transient faults (also known as soft errors) can appear as a data corruption in combinational or sequential parts of the circuit. If a high-energy particle hits the combinational parts of a circuit, a pulse will be produced which might be propagated into the system as well as to the final output. This phenomenon is usually termed as Single Event Transient (SET). In the case that the high-energy particle hits the sequential parts like flip-flops or SRAMs, it might lead to a bit-flip of the stored value, a so-called Single Event Upset (SEU). The rates of SET and SEU in each system are highly dependent on the relative amount of chip area devoted to combinational and sequential parts of the system [3]. Generally, by increasing the clock frequency, the errors observed will be dominated by SETs originating from combinational logic rather than SEUs on sequential logic [2], [4]. Moreover, relatively straightforward techniques exist to protect memory structures from SEU, while only few techniques exist for protecting the combinational parts of a system from SETs. Therefore, the effects of SETs in combinational logic networks are of particular concern in the transient fault community [3].

As an increased number of industrial domains use processors in safety-critical applications, such as X-by-wire systems in the automotive industry, the interest to understand the behaviour of processors in the presence of transient faults is increasing. Simulation-based fault injection is being used as a very detailed and accurate experimental way to demonstrate the efficiency of a processor in presence of transient faults. In simulation-based fault-injection, fault models are injected into the simulation model of a circuit described by one of the HDL languages [5]. This method has several advantages that make it very popular, i.e. it offers high controllability over where and when faults are injected. Furthermore, it provides high observability of faulty behaviours and fault propagation. Finally, it is feasible to carry out experiments during the design phase before the final hardware is being prototyped [5].

In recent years, much research has been published to study the effects of transient faults in processors by simulation-based experiments. Although there is still doubt how realistic these experiments are. The first concern rises from the accuracy of the processor model to conduct experiments. Are results still valid if experiments are done on a pre-synthesized HDL model compared to a detailed post-synthesized timing netlist? The second concern is about the number of fault injection experiments to get a realistic behaviour of the system under test. With a search in published works, one can easily find that different numbers of fault injection on a processor might lead to different behaviours of the processor for transient faults [2].

In this paper, the above-mentioned questions are approached by examining the effect of transient faults on a modern high performance reconfigurable processor, the Xentium® tile from Recore Systems [6]. Fault injection has been accomplished by using two models of this processor: a

[1] This research has been conducted within the CATRENE project TOETS (CT-302) which is financially supported by Agentschap NL.

978-1-4577-0375-1/11/$26.00 ©2011 IEEE

post placed-and-routed Verilog netlist (including timing information), and a pre placed-and-routed Verilog netlist. Moreover, the numbers of fault injection experiments are adjusted as far as convergence behaviour of the processor can be extracted. Our motivation to select the Xentium® processor is that the HDL description of this processor in different design phases (from abstract level to post place-and-route) was available. Secondly, this processor has been targeted to be used in automotive as well as space applications where an error can lead to loss of human life.

In this paper, we have focused on SETs on the datapath of the processor. A thorough study about SEUs on this processor will be discussed in another paper. Based on the results of SET injections, the vulnerable portions within the datapath are identified in order to drive some lightweight fault-tolerance mechanisms.

The rest of this paper is organized as follows: Section II surveys the recent works on simulation-based fault injection on processors. In Section III the core of the Xentium® processor and fault injection environment are described. In Section IV the results of the fault injection experiments are discussed. Section V concludes the paper and in section VI some ideas for future works are presented.

II. PREVIOUS WORKS

In recent years, a large number of simulation-based fault injection studies have been reported in literature. For example, [7] presents the impact of SEUs on the data cache memory of processors and in [8] the authors evaluate the susceptibility of different parts of a pipelined processor in presence of SEU. In [3] a deeply pipelined processor is targeted for SEU injection in order to detect its more vulnerable portion and then these portions have been protected by lightweight fault-tolerant mechanisms. Fault injection experiments are repeated on this improved architecture to assess the degree of resilience. In [2] a detailed analysis of the behaviour of a LEON2 processor that is equipped with Triple Modular Redundancy (TMR) is reported for both single and multiple SEUs. It is shown that the impact of multiple SEUs is identical to Single SEUs for high performance processors. In [9], the same version of the LEON2 processor has been targeted for fault injection but this time by *physical SEU injection*. While comparing these two papers, one can observe that the results from simulation-based experiments are challenging in comparison with those produced in *physical fault injection*. Considering that simulation-based experiments have been carried out without involving *timing information* of the model, we are inspired to use *timing information* of the netlist to define the importance of such information in simulation-based experiments. In [10], the authors discuss the behaviour of a Reconfigurable Compute Fabric (RCF) in presence of transient faults. In [11] the authors have proposed fault-tolerant Look Up Tables (LUT) for FPGAs and have assessed the dependability of this architecture by injecting single and multiple SEUs. As an industrial case, in [12] a dual core 32-bit System-on-Chip, which has been designed for automotive safety applications, has been injected with SEUs to validate the safety level of the architecture.

In most of the aforementioned works, the main objective has been to identify the most vulnerable parts of the targeted system for SEUs. To the best of our knowledge, there is no work focused on the importance of timing information of the netlist to produce accurate results. Furthermore, there is a gap in the knowledge about the effects of SETs on the combinational parts of processors that the current paper is addressing.

III. EXPERIMENTAL METHODOLOGY

In this section, first the processor architecture will be described. Next, the fault injection environment as well as the implemented workloads will be explained in detail.

A. Xentium Core Architecture

The Xentium® tile processor is designed by Recore Systems [6]. This processor is a reconfigurable processor that is being designed for high performance computing. The default architecture of a Xentium core including datapath, control unit, instruction cache, a network interface and memory banks is shown in Figure 1. The memory banks are SRAM memories that are communicating with the datapath in parallel to increase the parallelism. A detailed architecture of the datapath is shown in Figure 2. The datapath is designed based on VLIW (Very Large Instruction Word) architecture that consists of ten functional units and five register files. Each functional unit is responsible for a certain class of instructions. For example, E units (E0 and E1) perform load/store instructions; M, P, A and S units (M0, M1, P0, A0, A1, S0 and S1) perform arithmetic and logical operations. Finally, the C unit (C0) is dedicated to branch operations. All functional units can access five register files (RFA, RFB, RFC, RFD and RFE) in parallel.

A first fabrication of the Xentium® tile has been performed in 90nm CMOS technology leading to an area of 1.2mm^2 with a clock frequency of 200MHz. The implemented prototype is shown in Figure 3. This processor is developed as part of a multi-core SoC like depicted in Figure 4. Each tile can be connected to routers, while the routers in their turn are connected to a Network on Chip (NoC). The NoC can be connected to more conventional bus architectures to communicate with other peripherals, if required.

Given that one of our main objectives is to address the effects of *timing information* of the netlist on the validity of fault injection results, two netlists of the Xentium® tile have been used in this study. The first model is a Verilog netlist extracted *prior* to placed-and-routed without timing information (hereafter referred to as untimed netlist) while the second model is a Verilog netlist including timing information (hereafter referred to as timed netlist) which is extracted *after* placed-and-routed phase. Both of these netlists are based on the default architecture of the Xentium processor without any fault-tolerant mechanisms. As explained before, in this paper we look only at the combinational parts of the datapath to investigate the effects of SET-induced faults. The effect of SEUs on sequential parts is beyond the scope of this paper and will be discussed in another paper.

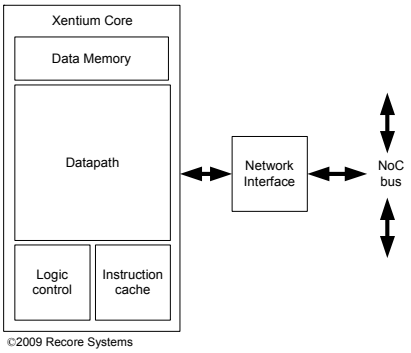


Figure 1. Xentium® processor with Memory and Network interface [6]

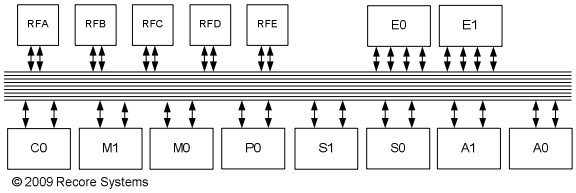


Figure 2. The Xentium® datapath architecture [6]

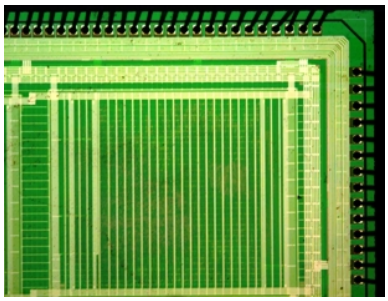


Figure 3. The fabricated Xentium tile in 90nm CMOS technology [6]

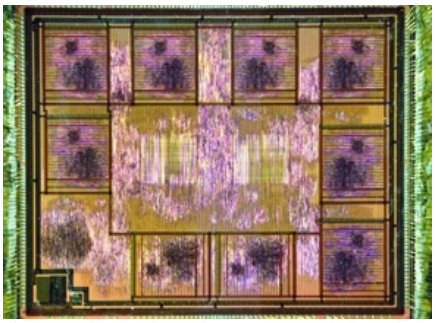


Figure 4. The manufactured multicore SoC consisting of nine Xentium® tiles [6]

B. Fault injection environment

A simulation-based fault injection has been chosen to study the effects of SETs on the Xentium® tile. Although this approach is time consuming, since simulation time is considerably longer than real-time execution, it has been employed in this paper because it allows verification of circuits early in the design phase. Moreover, verification via simulation can be performed in great detail since the designer has excellent control over when and where faults can be injected as well as a detailed monitoring of their consequence. In summary, the choice of simulation-based method allows very detailed analysis of the fault effects but limits the number of experiments due to its high computational requirements. In the following sections, it will be shown that by involving timing information of the netlist, the behaviour of the processor will converge after a number of fault injection experiments, so the designer can be satisfied with the number of experiments.

In our approach, the main fault injection mechanism is implemented via a minor code modification to inject transient pulses into combinational parts and alter their content at specified time, based on the idea of 'saboteur' [13]. As shown in Figure 5, the imported Fault Injection Entity (FIE) can inject positive pulses (Figure 5(a)) as well as negative pulses (Figure 5(b)) without interfering or affect in any way the other parts of the model. If the Fault Injection Signal (FIS) is inactive (zero), the victim signal will work as normal. However, If the FIS is active (high), a pulse will be injected in the victim net. The instant time and duration of the injected pulse is tightly related to the FIS signal. When the time in which the fault is to be injected is reached, the FIS will be high, so the OR gate (AND gate) masks the value of the victim signal and creates a positive (negative) glitch on the victim. When the duration of the injected pulse is past, the FIS returns to zero and the victim signal will be derived by the normal input. Characteristic of each injected pulse (time of occurrence and duration) is read by the simulator from a text file that is created before the beginning of simulations. It is important to note that this method makes a massive use of the Foreign Language Interference (FLI) of the commercial Verilog simulator (ModelSim®). It enables altering of FIS signals, monitoring and controlling the simulations, doing automatic file I/O handling whilst gathering data about the state of the processor after simulations.

In summary, the steps of the fault injection experiments are as follows:

1. Perform a golden run in order to obtain information such as total program execution time, contents of register files and memory banks at the end of the execution and the correct program output.
2. Generate a text file containing the instant injection time and duration for each FIS signal. The injection instant for all FIS signals is selected randomly with the lower bound five clock cycles (50ns) *after* the reset time (to inject the faults in the effective program execution time) and the upper bound five clock cycles (50ns) *before* the total program execution time (to give enough time to faults to propagate in the system). The duration of each FIS is selected in such a way that all FIS signals that have been targeted be a victim compose an exponential distribution with a mean value μ, as close as possible to the clock duration. The reason to select the exponential distribution is inspired by the fact that this distribution creates the highest rate of effective faults with regard to the clock cycle, compared to other conventional distributions such as the Normal and Poisson distributions. A detailed analysis on the effects of different distribution functions on fault experiments has

been carried out by the authors and has been discussed in a separate paper [14].

3. Run the simulations. In each experiment, only one fault is being injected. Therefore, the simulator restarts the simulation, and runs the simulation until it reaches the fault injection time. Then the victim signal is altered based on the duration of FIS signal. The FIS signal will be deactivated after the duration time and simulations continues for a specified time (20 clock cycles after program execution time). At the end, the value of the primary output as well as all register files and memory banks are stored as a text file in the local hard drives.
4. Analyze the stored data by comparing the golden run results with the results of fault injection experiments.

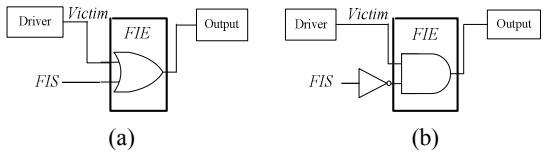


Figure 5. Injection of SET. (a) Positive glitch. (b) Negative glitch

C. Workloads

To involve several parts of the Xentium® processor in fault injection experiments, a number of different benchmarks from a specific domain have been used. The benchmarks have been selected from the Mibench benchmark suits [Automotive and Industrial Control domain] [15]. The first benchmark is a Qsort program that sorts a large array of strings into ascending order using the well-known quick sort algorithm; the second is a BitCount program that tests the bit manipulation abilities of a processor by counting the number of bits in an array of integers. The input data is an array of integers with equal numbers of 1's and 0's; finally, the BasicMath program that performs calculations that often do not have dedicated hardware support in embedded processors. For example, cubic function solving and integer square root. The input data is a fixed set of constants. The workloads chosen were selected for their diversity within a specific domain, as our focus is on the automotive arena. Figure 6 shows the dynamic instruction distribution of each workload. This chart will reveal interesting aspects of workloads in the case that fault injection results will be analysed.

Each mentioned benchmark was written in the assembly language of the Xentium and subsequently converted in machine code by Xentium assembler. The machine code can be directly programmed in the instruction memory of the Xentium processor.

As mentioned before, only SET faults are being considered in this paper. In each experiment, a single SET fault is injected in the datapath. The probability of one functional unit being selected depends on the area of that unit. The effect of faults is divided as follows:

- No effect: this means that the program terminates in a normal way and the results are identical to those in the golden run.
- Failure: the program terminates normally but the results of the final output are not identical to the golden run.

It is important to note that there are other scenarios for the effect of faults, for example, the program never terminates normally (and should be halted by Modelsim instructions) and the SDCs (Silent Data Corruptions); However, these effects are eliminated from the overall results due to their negligible contribution (less than 1%).

Because an exhaustive number of experiments are not feasible in this type of complex system, the number of experiments starts from a small amount of numbers and subsequently the numbers are increased until convergence behaviour of the processor can be identified. For the first set of experiments, we started with 500 simulations and for the next step the number of experiments has been increased to be confident that the final behaviour of the processor to SETs is independent of the number of fault injection experiments.

IV. FAULT INJECTION RESULTS

This section is divided into two subsections. First, the general behaviour of the datapath for SETs is explained and next, the contribution of each functional unit to produce failures is dealt with.

A. General behavior

As mentioned before, fault injection experiments have been carried out on two models of the processor, a timed netlist and an untimed netlist. In order to get an overall view of the datapath behaviour, the total probability theorem can be used. The total failure rate of the datapath (P_{total}) can be calculated by:

$$P_{total} = (A_E/A_{total})\,P_E + (A_S/A_{total})\,P_S + (A_A/A_{total})\,P_A + (A_C/A_{total})\,P_C + (A_P/A_{total})\,P_P + (A_M/A_{total})\,P_M \quad (1)$$

Where A_E, A_S, A_A, A_C, A_P, A_M are the area of each functional unit, A_{total} is the total area of the datapath and P_E, P_S, P_A, P_C, P_P, P_M are the failure rates for each functional unit measured directly from the simulation results.

The general behaviour of the datapath for each netlist is shown in Figures 7(a) and 7(b). The different behaviour in performance of the processor for timed and untimed netlist is clear from Figure 7. First, a lower rate of injected faults has been propagated as failures in the untimed netlist (about one-third as compared to those in the timed netlist). Moreover, it is very important to note that finding convergence behaviour of each netlist is very crucial to judge the performance of the processor.

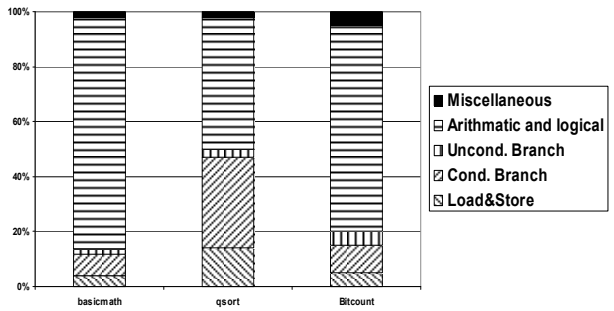
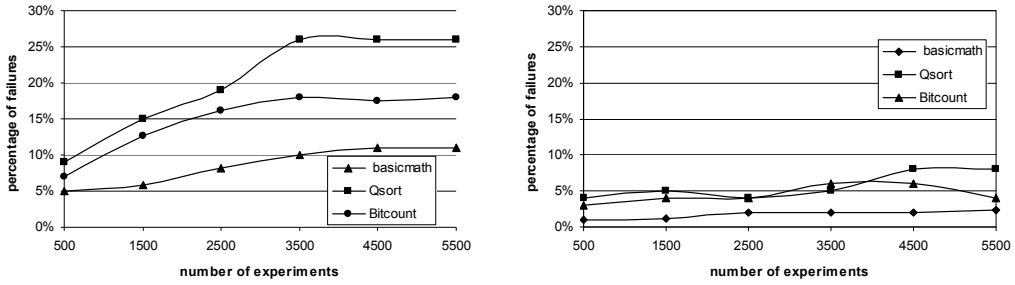


Figure 6. Instruction distribution for each workload



(a) (b)

Figure 7. Behavior of two netlists: (a) timed netlist, (b) untimed netlist

In the timed netlist, the results nicely convergence beyond 3500 experiments and this convergence point is independent of the workloads. Hence, the behaviour of the processor for the 3500 experiment can be considered as a stable behaviour of the Xentium® with regard to SETs. While in the untimed netlist, it is very hard to find a distinguishable point of convergence for all experiments. For example, in the BasicMath program, the performance of the processor is quite stable from 500 to 5500 experiments, while in the BitCount program, the processor has a deviating performance without any reasonable convergence which makes it difficult to define its behaviour.

As a summary, with the involvement of timing information of the model and the appropriate number of experiments, the designer can be convinced with respect to the number of experiments that there is sufficient confidence to judge about the processor's behaviour. The trade-off to involve timing information is that the Standard Delay Format (SDF) back annotation for each run is very time consuming. A new platform to reduce the computational complexity of simulation-based fault injections has been carried out by the authors and has been discussed in [14]. In the current paper, the behaviour of the processor for 3500 experiments on the timed netlist has been considered as convenient behaviour.

From Figure 7(a), several aspects of the processor's behaviour can be discovered. First, the dependence of the performance of the processor on the workload is clear. It can be seen that the Qsort program has the worst performance in producing failures. This is due the fact that this program has the highest portion of memory–related instructions and at the same time the highest portion of branch-related instructions (Figure 5). Later on, it will be shown that the E and C0 units (responsible for load/store and branch instructions, respectively) are the most sensitive parts of the processor with regard to SETs. Hence, the extensive use of load/store and branch instructions in a workload will obviously raise the probability to generate failures.

The improved rate of failures observed in the BitCount and BasicMath programs are due to the fact that these workloads are more computational extensive with fewer memory references (as can be seen in Figure 6). The BitCount benchmark uses a number of memory-related instructions to load an array of numbers from the memory. It then uses ALU instructions to compute the number of ones in that array, resulting in a better performance as compared to the Qsort program. In the BasicMath workload the portion of memory-related instructions is much smaller and the proportion of ALU-related instructions higher, resulting in the best performance among all workloads.

B. Functional units contribution

Figure 8 depicts the contribution of each functional unit to the overall datapath behaviour for the different workloads. These statistics have been achieved for 3500 experiments on the timed netlist; as was mentioned before, this number of experiments is sufficient to judge the behaviour of the processor. The following conclusions can be extracted from Figure 8.

The first observation concerns the high portion of the E unit to produce failures. An important issue is that the E unit has the most portions for all three workloads. It means that independent of which workload is running, the E unit is the most sensitive part of the datapath with regard to SETs. Especially if we note that the E units has been involved in 7.6% of instructions (in average of all benchmarks) while they have contributed about 29% to the overall failure rate. So this unit should definitely be protected by circuitry or architectural methods to decrease the probability of failures. The second observation concerns the importance of the A and C units to the overall failure. It is not straightforward which unit contributes mostly to failures, as it is dependent on the workload. In the BasicMath workload which is highly computational, the A units contribute more to the generated failures as compared to the C0 unit. In the BitCount workload, the C0 unit has a higher contribution with regards to failures. The third observation is the workload-independent behaviour of the S, P and M units. While the average percentage of usage of these units in the all three benchmarks is relatively high (about 69%), they have a negligible contribution to the overall failures.

In summary, by running benchmarks with different distributions of instructions, several interesting results can be extracted regarding the behaviour of the processor. However, it is important to note that we are primarily interested in a specific domain of applications, which is the automotive arena in our case.

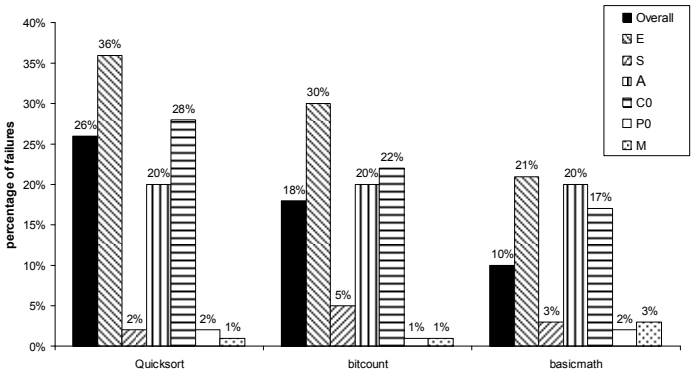


Figure 8. Contribution of different parts of functional units for different benchmarks

V. CONCLUSIONS

In this paper, we have addressed several aspects that play crucial roles in transient fault studies. First, the timing information of the model has a key role in perception of the behaviour of the processor regarding transient faults. By carrying out experiments at the top-level extracted model without knowledge about the final timing information can lead to a different performance that is highly dependent on the number of experiments. The second observation is that despite the conclusion of some papers which have studied the impact of SETs on untimed netlists, the effect of SETs on sub-micron technologies is not negligible at all. Especially if we note that timing characteristics of future technologies are intended to highlight the importance of SETs in combinational circuits. The third observation is that the workloads within a domain should provide enough diversity to involve different parts of the processor. Considering the mentioned notes, we have performed a detailed study to investigate the effects of SETs on the combinational parts of a modern high performance processor. The results showed that approximately 15% of SETs on the datapath of the Xentium processor would lead to a catastrophic failure.

VI. FUTURE WORKS

Several ideas can be derived as the future of this paper. First, the results of this study will be used to propose some lightweight mechanisms to protect the most vulnerable parts of the Xentium in presence of SETs. Second, is the comparison of simulation-based results with silicon fault injection experiments. As the final chip is available at this moment, we are able to conduct physical fault injection on the fabricated Xentium tile with the same workloads, to assess the degree of reality of simulation-based fault injection on the timed netlist.

ACKNOWLEDGMENTS

We would like to thank Dr. Gerard Rauwerda and Sebastien Baillou from Recore Systems for the support with the Netlists and Verilog codes.

REFERENCES

[1] R. C. Baumann, "Soft Errors in Advanced Semiconductor Devices–Part I: The Three Radiation Sources," *IEEE Trans. on Device and Material Reliability*, vol. 1, no. 1, pp. 17–22, 2001.

[2] E. Touloupis, J. A. Flint, V. A. Chouliaras and D. D. Ward, "Study of the effects of SEU Induced Faults on a pipeline Protected Microprocessor," *IEEE Trans. on Computers*, Vol. 56, no. 12, pp. 1585 – 1596, 2007.

[3] N. J. Wang, J. Quek, T. M. Rafacz and S. J. Patel, "Characterizing the effects of Transient Fautls on a High Performance Processor Pipeline", *International Conf. on Dependable Systems and Networks*, pp. 61-70, 2004.

[4] Irom and F. F. Farmanesh, "Frequency Dependence of Single-Event Upset in Advanced Commercial PowerPC Microprocessors," *IEEE Trans. on Nuclear Science*, vol. 51, no. 6, pp. 3505–3509, 2004.

[5] J. C. Baraza, J. Gracia, D. Gil, P. J. Gil, "Improvement of fault injection techniques based on VHDL code modification," *IEEE High-Level Design Validation and Test Workshop*, pp. 19-26, 2005.

[6] Recore Systems, "www.Recoresystems.com"

[7] F. Faure, R. Velazco, M. Violante, M. Rebaudengo, and M. S. Reorda, "Impact of Data Cache Memory on the Single Event Upset-Induced Error Rate of Microprocessors," *IEEE Trans. on Nuclear Science*, vol. 50, no. 6, pp. 2101–2106, 2003.

[8] M. Rebaudengo, M. S. Reorda, and M. Violante, "Accurate Analysis of Single Event Upsets in a Pipelined Microprocessor," *Journal of Electronic Testing: Theory and Applications*, vol. 19, no. 5, pp. 577–584, 2003.

[9] M. Rebaudengo, M. S. Reorda and M. Violante, "Accurate Analysis of Single Event Upset in a Pipelined Microprocessor," *Journal of Electronic Testing: Theory and Application*, Vol. 19, no. 5, pp. 577-584, 2003.

[10] L. Sterpone and M. Violante, "Dependability evaluation of transient fault effects in Reconfigurable Compute Fabric Devices," *in the Proc. of IEEE International On-line Testing Symposium,* pp. 2-8, 2006.

[11] H. R. Zarandi, S.G. Miremadi, C. Argyrides, D. K. Pradhan, "Multiple SEU Tolerance in LUTs of FPGAs Using Protected Schemes," *In the Proceedings of 12th IEEE European Test Symposium*, Germany, 2007.

[12] O. Bailan, U. Rossi, A. Wantens, J. Daveau, S. Nappi and P.Roche, "Verification of Soft Error Detection Mechanism through Fault Injection on Hardware Emulation Platform," 4th *Workshop on Dependable and Secure Nanocomputing (WDSN)*, pp. 113, 2010.

[13] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. Karlsson, "Fault Injection into VHDL Models: The MEFISTO Tool," *In Proceedings of the 24th Annual International Sym. on Fault-Tolerant Computing*, pp. 66–75, 1994.

[14] A.Rohani, H.G. Kerkhoff, "A Technique for Accelerating Injection of Transient Faults in Complex SoCs," *accepted for the 14th Euromicro Conference on Digital System Design (DSD)*, 2011.

[15] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," *In IEEE Workshop on Workload Characterization*, pp. 3-14, 2001.