A software solution to estimate the SEU-induced soft error rate for systems implemented on SRAM-based FPGAs*

Wang Zhongming(王忠明)[1,2,†], Yao Zhibin(姚志斌)[2], Guo Hongxia(郭红霞)[2], and Lü Min(吕敏)[1,2]

[1]Department of Engineering Physics, Tsinghua University, Beijing 100084, China
[2]Northwest Institute of Nuclear Technology, Xi'an 710024, China

Abstract: SRAM-based FPGAs are very susceptible to radiation-induced Single-Event Upsets (SEUs) in space applications. The failure mechanism in FPGA's configuration memory differs from those in traditional memory device. As a result, there is a growing demand for methodologies which could quantitatively evaluate the impact of this effect. Fault injection appears to meet such requirement. In this paper, we propose a new methodology to analyze the soft errors in SRAM-based FPGAs. This method is based on in depth understanding of the device architecture and failure mechanisms induced by configuration upsets. The developed programs read in the placed and routed netlist, search for critical logic nodes and paths that may destroy the circuit topological structure, and then query a database storing the decoded relationship of the configurable resources and corresponding control bit to get the sensitive bits. Accelerator irradiation test and fault injection experiments were carried out to validate this approach.

Key words: radiation effect; single-event effect; SRAM-based FPGAs; fault injection
DOI: 10.1088/1674-4926/32/5/055008 **EEACC:** 2550; 2530; 2570

1. Introduction

Reconfigurable SRAM-based FPGAs have already become an appealing solution for space electronic equipments. However, the large amount of SRAM cells brings high sensitivity to radiation-induced Single-Event Upsets (SEUs). When considering the SEU effect in FPGAs, two kinds of upsets should be considered respectively. The first one happens in the embedded user memory, which can be viewed as a temporary effect. The disruption disappears as soon as a new value is written into the affected element. The second type of SEU happens in the configuration memory, which is much severer in practice. They can permanently modify the topological structure of the user circuit until the bitstream is reloaded. Fault tolerant design techniques, such as triple modular redundancy (TMR) and periodic scrubbing, were suggested to mitigate this kind of upset[1,2].

Generally, the SEU tolerance is characterized using the concept of upset cross section. The cross section varies as a function of the impinging particle's linear energy transfer (LET) or proton's energy, which can be fitted by a Weibull curve. The parameters of this curve can be used as input of programs like CREME96 to predict the system failure rate in different orbits. In radiation test, the device static cross section is computed as the upset number divided by the incident particle fluence.

$$\sigma_{\text{static}} = \frac{\text{Number of Upsets}}{\text{Particle Fluence}}. \quad (1)$$

However, a design never spends all the configurable resources in a specific device. As a result, not all upsets have impact on the designed system. If the device upset cross section is used to predict the system failure rate, the result will be too conservative. The concept of dynamic cross section is suggested to estimate the system dependability in SRAM-based FPGAs, in which the number of upset is replaced by the number of system failures[3].

$$\sigma_{\text{dynamic}} = \frac{\text{Number of System Failures}}{\text{Particle Fluence}}. \quad (2)$$

On the other hand, if we define "sensitive" bits as those which may induce modifications to the implemented design, the difference between system failure and configuration upset would be equal to the number of sensitive bits over the total number of configuration bits[4], as shown in the following equation:

$$\epsilon = \frac{\sigma_{\text{dynamic}}}{\sigma_{\text{static}}} = \frac{\text{Number of Sensitive Bits}}{\text{Number of Configuration Bits}}. \quad (3)$$

The static cross section should only be characterized using accelerator irradiation test. However, if different systems built on FPGAs have to be verified using accelerator test, it would be too expensive and unpractical. Therefore, simulation approaches are developed to evaluate the dynamic SEU effect for a specific design.

Several fault injection systems have been reported[5−7]. Generally, a reference device is needed and the output mismatch of the two device is monitored. The only difference between fault injection and real SEUs is that the upsets are not introduced by real particles but artificial modification of the

* Project supported by the National Natural Science Foundation of China (No. 10875096).
† Corresponding author. Email: wang-zm02@mails.tsinghua.edu.cn
Received 8 November 2010, revised manuscript received 4 December 2010 © 2011 Chinese Institute of Electronics

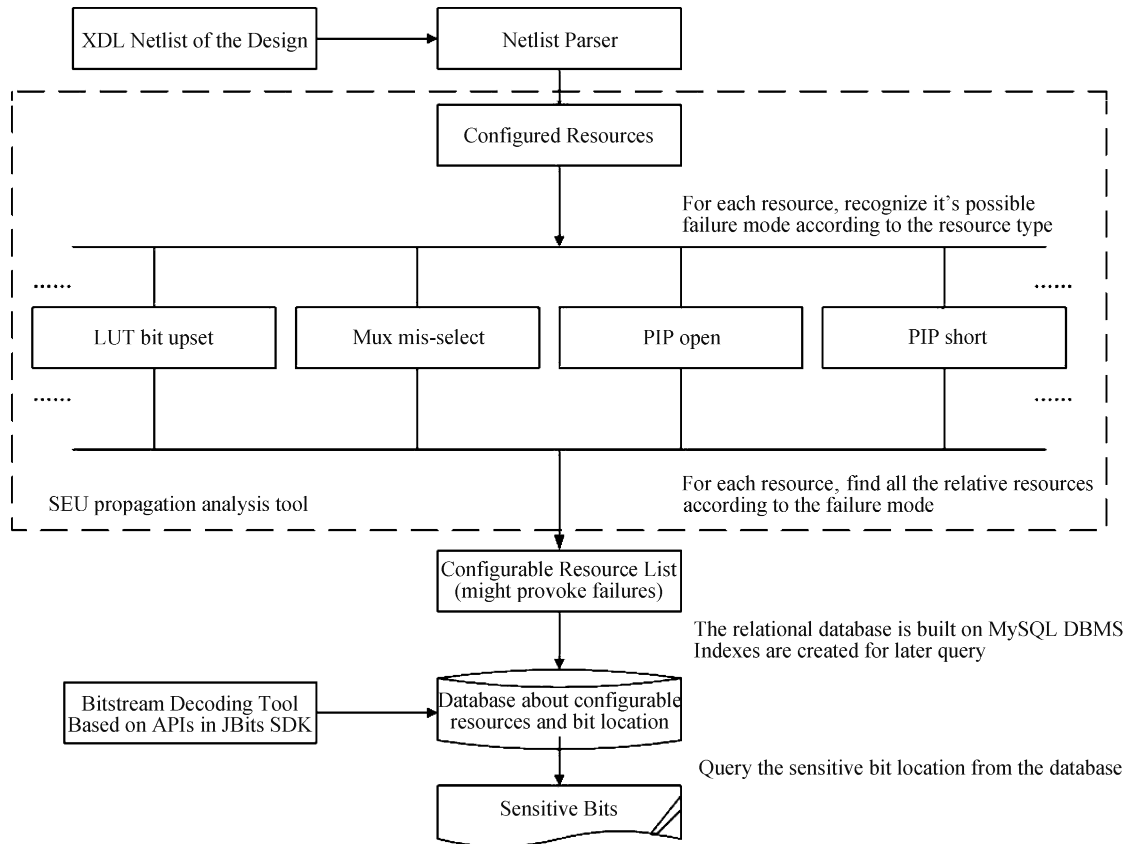

Fig. 1. The flow chart of the proposed analysis tool.

bitstream. Bits are flipped one by one before written into the configuration memory. Fault injection is very attractive for its great flexibility and has become a standard tool to study the effectiveness of redundant mitigation methods[8, 9]. However, there are still some drawbacks of fault injection. First, it treats the device as a black box and there is no prior information about where the sensitive bits may locate and how they affect the system behavior. As a result, the injection procedure is time consuming. Second, the injection result does not contain enough information for improving the mitigation strategies. Moreover, if the look up table (LUT) is configured as user memory, the bit injection into these places may alter current circuit state and cause persistent output error, which would interrupt the automatic injection flow.

There are also some other simulation approaches besides fault injection. STAR-LX is proposed in a series of works[10−12], in which a graphic model of the FPGA architecture is developed, and then an algorithm is used to search for sensitive nodes in the graph. This program only deals with the routings of the device. The failures inside a logic node is not mentioned. Another tool named STARC was also designed to analyze the reliability issue in FPGAs with the emphasis of domain cross errors in TMR structure[13]. It uses the industry standard EDIF circuit representation as input and a dependency graph of the circuit is created during a hierarchical exploration. The limitation of this method is that EDIF does not contain placement and routing information. STARC has to use statistical model to estimate the sensitive bit for each routing logic. By the way, these two projects are kept as a closed property and not available for public usage. The drawbacks of existing tools drive us to find alternative solution to assess the reliability issue of systems build on SRAM-based FPGAs.

2. Overview of the proposed methodology

After a comprehensive review of literatures, we realize that if we can model FPGAs in different hierarchies using objective orient manner and achieve information about which and how the resources are programmed, the critical part of the circuit might be located more directly.

The developed approach is composed of three modules, as shown in Fig. 1. The first part is a bitstream decoding tool. The decoded result is the relationship of programmable resources and corresponding control bits. The second part is a netlist parser, which reads in the placed and routed netlist to get all the configuration state of programmed resources. The third part defines an SEU propagation rule. For each configured resource, we identify all the relative logic nodes and paths, which may destroy the circuit topological structure. The control bits of these critical resources are considered to be "sensitive". If upset occurs in these places, the circuit description might be affected. In the following sections, we will describe how each part are implemented to constitute our analysis tool.

2.1. Basic structure of the Xilinx FPGAs

Xilinx FPGAs have a regular array of configurable logic blocks (CLB), surrounded by configurable I/O blocks (IOB) and on-chip user BlockRAMs[14,15], as shown in Fig. 2. The CLBs are programmed to implement different digital logic

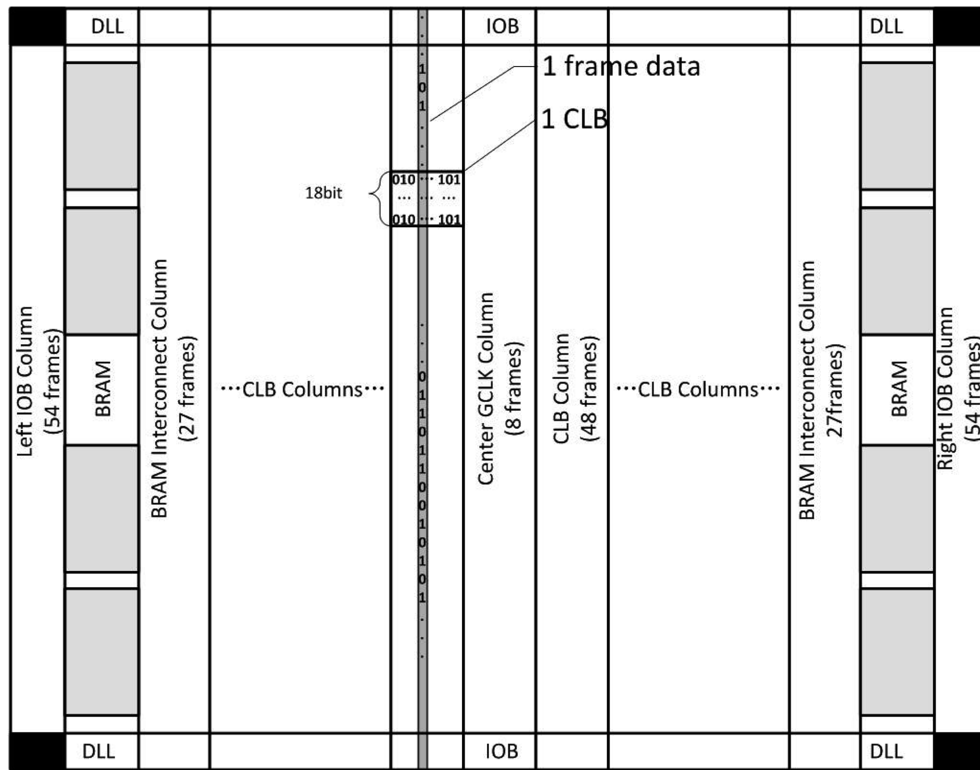

Fig. 2. The simplified architecture and bitstream organization manner in Xilinx FPGAs.

and sequential functions. The IOBs provide interfaces between CLB and package pins. Figure 3 provides a schematic view into a Slice with the alphabets emphasis on the programmable bits. In addition to the resources inside the Slice, there are also large amount of programmable routings around the Slices in CLB.

The CLB routing resource is divided into two parts: the input/output Mux and the general routing. The input/output Mux is used to select input/output pins to the general routings, while the general routings can be programmed to form custom signal transport lines between CLBs. The wires are each grouped into busses that extend in the four primary directions. The connections to neighboring CLBs are straightforward, but two wires from different directions may have a programmable interconnect point (PIP), which can be open or closed according to the control bit. The PIPs constitute a switch matrix that allows wires from different directions to be connected[16].

2.2. Bitstream decoding using JBits

The association between programmable resource and corresponding bit location needs a detailed knowledge of the FPGAs architecture and its configuration bitstream. The configuration memory can be visualized as a rectangular array of bits. The bits are grouped according to its corresponding resource column, as shown in Fig. 2. One-column bits are further divided into several one-bit wide frames. Each frame is numbered with an index according to certain rule and then sequentially arranged into the bitstream according to that index. If the relative location of a bit inside a CLB (or IOB) is known, the bit address in the whole bitstream can be calculated.

The problem is how to get the relative bit location of a certain resource. One of the decoding method was described in literatures[17, 18]. It is based on JBits, which contains a set of application program interfaces (APIs) to set or probe the state of the programmable resources in Xilinx FPGAs. The resources are modeled using Java classes in JBits. Therefore, it provides the lowest level interfaces into the device architecture. The basic design flow of JBits is to read in an original bitstream, modify portion of the circuit and generate a new bitstream. We can modify the value of a certain resource once at a time, generate the modified bitstream and then compare it with the original one. This procedure is repeated until all the resources in one CLB or IOB are identified. The result can be easily generalized to the whole device according to the regular structure of FPGAs.

2.3. Netlist parser

Another important issue is to find the resources used in a specific design. The placed and routed netlist is stored in Native Circuit Description (ncd) file in Xilinx EDA tool chain. It contains the structural and layout description of the circuit. The ncd file has a closed binary format. Xilinx provides another tool named xdl, which could convert ncd file into an ASCII text file written in Xilinx design language (xdl).

The xdl netlist is quite easy to understand. There are two kinds of object in xdl netlist: instance and net. Instance describes the configuration state of employed CLBs and IOBs. The configuration string contains information of how the internal LUT and control logic are used inside a Slice. A net in the xdl netlist corresponds to a real connection line in the design, which contains an input pin, one or more output pins and several PIPs in the routing matrix. A netlist parser is developed to get the information about which and how a Slice or routing

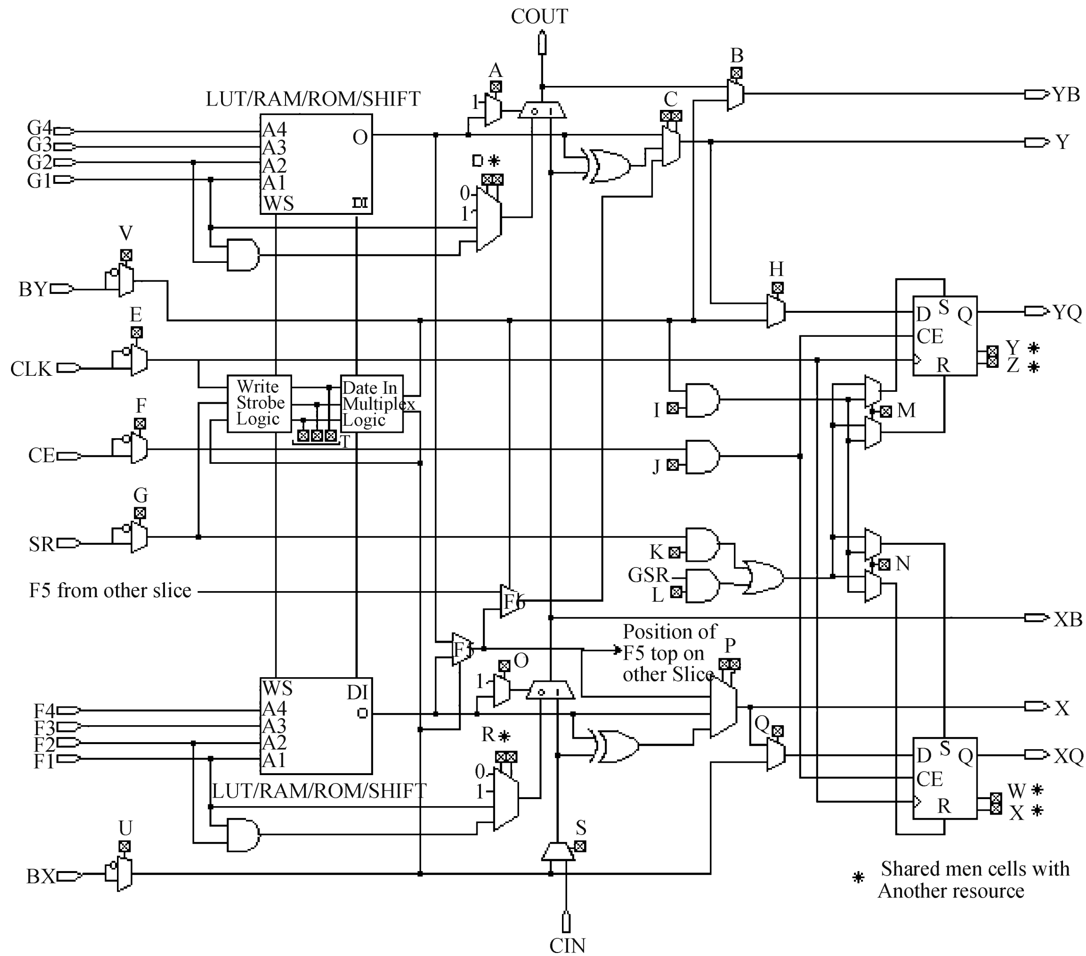

Fig. 3. The internal structure and the configurable resources inside the Slice, viewed from the FPGA Editor.

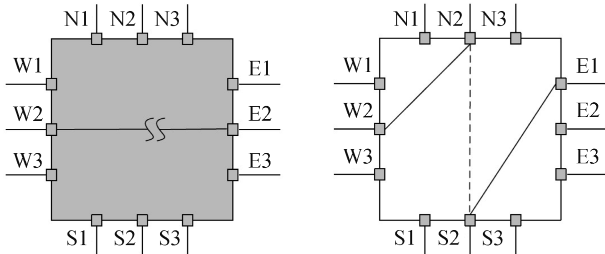

Fig. 4. A sketch map of the PIP open and short cut failure.

structure is programmed.

2.4. SEU propagation rule

The elements used in a certain design are available after the placed and routed netlist is parsed. However, these elements are not the only sensitive resources according to the failure mechanisms in FPGAs[19−21]. It is worth mentioning that the SEUs in adjacent elements may also provoke contentions to the circuit structure. The SEU propagation rule will discuss how different types of resource are treated to find relative critical elements.

The logic functions in Xilinx FPGAs are implemented inside CLB Slices. A truth table is stored in LUTs and the value is read out according to the combination of the inputs. The first kind of failure in FPGAs is the bit flip in LUTs. The configuration string in the instance of xdl netlist gives a detailed description of which LUT is employed and the logic function it implements. All the employed LUT bits are considered sensitive.

The second type of failure comes from the control bits inside the CLBs or IOBs as shown in Fig. 3. They perform important functionality, such as internal Mux selection and LUT functional control. When upset happens in these places, the impact is unpredictable. The configured detail is also included in the configuration string of the instance. The Slice internal control logic and routings whose state is not “OFF” are considered to be sensitive.

Typically, the routings outside Slices constitute over 80% of the configuration bits, among which the largest contribution comes from the input, output and hex multiplexer. The multiplexer and corresponding control bits is not a one-to-one relationship. For example, a Mux with four sources and one sink is likely to be controlled with only two bits. Any upset in these two bits may lead to a different connection. When considering the SEU effect, all the bits in charge of one Mux should be considered sensitive.

A PIP in the Single Switch Matrix of the General Routing is controlled exactly by one bit, which differs from multiplexer. When a PIP is found in the netlist, there is a connection between two existing lines. If this bit is flipped by SEU, an open failure occurs. As a result, these PIP control bits are classified

as sensitive. The left figure in Fig. 4 is a sketch map of this situation.

There is another possibility that a PIP is off at the beginning and it happened to bridge two existing signal lines after opened by a flip, as the right figure in Fig. 4. We developed an traversal algorithm to search for this kind of short cut. Note that, this type of PIP is not included in the netlist at the beginning.

3. Development and validation of the proposed method

The proposed approach has been developed as a software project and validated using fault injection and accelerator irradiation test.

3.1. Overview of the project

The SEU analysis tool is written in objective-oriented way to model actual structures inside Xilinx FPGAs. The whole project is composed of over 40 classes in Java in order to get seamless integration with JBits. The relationship between configurable resources and bitstream addresses is stored and managed using a MySQL database. The client program read in the placed and routed netlist, parse it to get the utilized resources and their configuration states, treat them separately to get the critical elements according to the SEU propagation rule, then communicate with the database to get the sensitive bits. The project is now compatible with several Xilinx device series. The flow chart of the SEU analysis tool is shown in Fig. 1.

3.2. Development of validation platform

The effectiveness of the developed tool need to be validated. A radiation test and fault injection hardware platform was built. It is composed of three parts, a control board, two test boards and a controlling GUI program. The internal memory upset is monitored through periodic readback. The test boards contain two identical FPGAs. When a dynamic test mode is chosen, the two devices are programmed with the same design and run synchronously. The outputs of these two devices are being monitored continuously. The device under test (DUT) is irradiated under accelerator beam and the other golden device runs as a reference. As soon as a discrepancy is detected, a functional failure is recorded and the device is reconfigured automatically.

The fault injection system is built on the same hardware. Partial reconfiguration technique of advanced Xilinx FPGAs is exploited to inject upset into the configuration memory. One bit of the injected frame has already been flipped before downloaded, and the output is being monitored until the test vector runs to the end. If functional failure occurs, the injected bit is classified to be sensitive.

3.3. Benchmark design

Three benchmark circuits are designed for the dynamic test. The first design B01 is a simple 16-bit latch, which uses less than 1% of the resources. It was originally designed to verify the test system. The second benchmark B02 is a 1024×16 bit shift register chain, which uses about 16% of the entire flip flops, but without any LUTs. This design is meant to detect the upset of flip flops. The third design B03 is a resource balanced one, which uses three kinds of IP cores: two linear shift registers, two multiply accumulators and two FIFOs. This design occupies about 12% of the LUTs and 31% of the Flip Flops.

3.4. Experiment setup

Xilinx Virtex XCV300 is adopted as DUT in our experiment. This device is manufactured in a 0.22 μm commercial fabrication technology with over 0.3 M gates and 1.5 M bits of SRAM cells in the configuration memory. It has been widely applied both in industry and research field.

The experiment was carried out at HI-13 Tandem Accelerator in Beijing. Device was irradiated under 175 MeV Cl ions with an LET of 12.6 MeV/(cm^{-2}·mg). During the static test, the bitstream was read back to detect errors. During the dynamic test, the system function is monitored continuously. Once a functional failure occurs, the device is reprogrammed automatically to fix the upset and the fluence is recorded. The ion flux was set to a relatively low level (about 200–1000 (cm^2·s)$^{-1}$) to prevent failures from happening too fast. Note that the functional errors are less than the upsets, especially when the design resource utilization is not severe. Therefore, sometimes it is rare to see. We decided to accumulate about 30 errors due to the time limitation of the beam.

Fault injection is also deployed to these benchmarks. During the injection, we modified the bitstream using partial reconfiguration technique. After injecting one bit at a time, a test vector is generated using random numbers. The output is monitored using another identical FPGA running the same design synchronously. Once a mismatch is found, the bit is identified to be sensitive. About 1.5 h is needed to complete an exhaustive injection flow.

Finally, the code developed in this paper runs to analyze the sensitive bits in these three designs. After the optimization of the database for unique query demand, the analysis tool runs at a very fast speed. It takes only about 20 min, 5 min and less than 1 min to finish the three cases on an ordinary personal computer.

3.5. Test result and discussion

The result of radiation test is listed in Table 1. Equation (1) is used to calculate the static upset cross sections and Equation (2) for the dynamic cross sections. The coefficient ϵ is calculated using the ratio of dynamic and static cross sections here. It is the probability that the system will be affected when the configuration memory is randomly flipped by one bit. The dynamic cross section can be estimated from the static cross section and ϵ.

In fault injection and software analysis, ϵ is calculated using the number of sensitive bits divided by the number of total configuration bits, as Eq. (3). The experiment result is listed in Table 2. The coefficient of fault injection and radiation test is close, but 1.x times higher in software analysis. This result is mainly because the defined SEU propagation rule is too strict in some situations. It will detect every single bit which might provoke disturbance to the user circuit. However, due to the limitation of the test vector and extra control bits of a multiplexer, the analysis results tend to be conservative in practice.

Table 1. The static and dynamic test result of the three benchmark circuits.

Design	Fluence	Functional errors	Static cross section	Dynamic cross section	Coefficient ϵ
B01	1.00×10^6	24	3.11×10^{-2}	2.40×10^{-5}	7.72×10^{-4}
B02	1.16×10^5	31	3.11×10^{-2}	2.67×10^{-4}	8.59×10^{-3}
B03	6.97×10^4	107	3.11×10^{-2}	1.54×10^{-3}	4.94×10^{-2}

Table 2. The fault injection and software analysis result of the three benchmark circuits.

Design	Configuration bits	Injection errors	Coefficient ϵ	Sensitive bits of analysis	Coefficient ϵ
B01	1.47×10^6	894	6.27×10^{-4}	899	6.31×10^{-4}
B02	1.47×10^6	17000	1.19×10^{-2}	26768	1.82×10^{-2}
B03	1.47×10^6	69870	4.90×10^{-2}	133939	9.10×10^{-2}

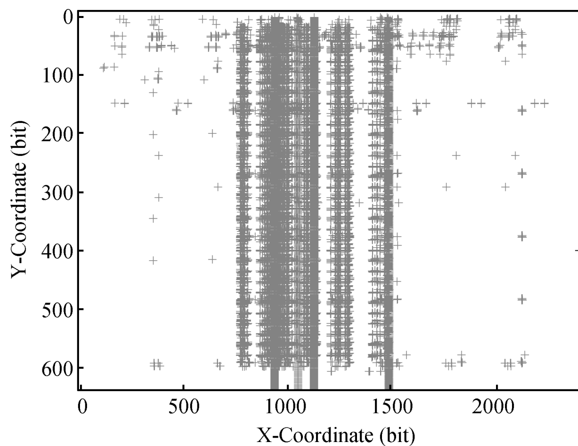

Fig. 5. Mapping the sensitive bits from fault injection to its real location.

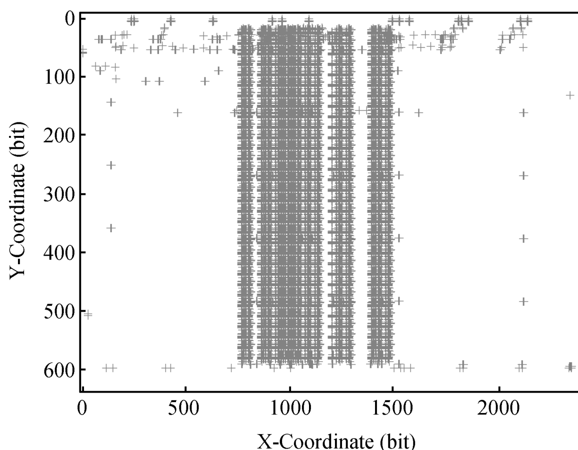

Fig. 6. Mapping the sensitive bits from netlist analysis to its real location.

The coefficient ϵ reflects the statistical information of the sensitive bits. The bit addresses are also available both in fault injection and software analysis. We made a further comparison of these details in an intuitive way. The physical location of the sensitive bits of benchmark B02 is plotted in Fig. 5 and Fig. 6 using the coordinate transformation rule described in Xilinx Application Notes. As we can see from these two figures, most of the sensitive bits in fault injection are also identified by the software analysis. The analysis tool provides an upper limit of the sensitive bits.

4. Conclusion

In this paper, we developed a speedy and handy tool for system engineers to assess the reliability of designs implemented on SRAM-based FPGAs. The central idea is to find the critical placement and routings from the netlist, so that it could avoid traversing the bit locations and test patterns as traditional fault injection. The result can be used to predict the Single-Event induced soft error rate in space applications.

References

[1] Xilinx Inc. Triple module redundancy design techniques for virtex FPGAs, Xilinx application note XAPP197. Version 1.0.1, 2006, http://www.xilinx.com/ support/ documentation/application notes/

[2] Xilinx Inc. Correcting single-event upsets through virtex partial configuration, Xilinx application note XAPP216. Version 1.0, 2000, http://www.xilinx.com/support/documentation/application notes/

[3] Morgan K, Caffrey M, Graham P, et al. SEU-induced persistent error propagation in FPGAs. IEEE Trans Nucl Sci, 2005, 52(6): 2438

[4] Violante M, Sterpone L, Ceschia M, et al. Simulation-based analysis of SEU effects in SRAM-based FPGAs. IEEE Trans Nucl Sci, 2004, 51(6): 3354

[5] Johnson E, Wirthlin M, Caffrey M. Single-event upset simulation on an FPGA. Engineering of Reconfigurable Systems and Algorithms (ERSA), Las Vegas, Nevada, USA, 2002

[6] Alderighi M, Casini F, Angelo S, et al. A tool for injecting SEU-like faults into the configuration control mechanism of Xilinx Virtex FPGAs. 18th IEEE Defect and Fault Tolerance in VLSI Systems, 2003

[7] Johnson E, Caffrey M, Graham P, et al. Accelerator validation of an FPGA SEU simulator. IEEE Trans Nucl Sci, 2003, 50(6): 2147

[8] Morgan K, McMurtrey D, Pratt B, et al. A comparison of TMR with alternative fault-tolerant design techniques for FPGAs. IEEE Trans Nucl Sci, 2007, 54(6): 2065

[9] Alderighi M, Casini F, Weigand S, et al. Evaluation of single-event upset mitigation schemes for SRAM-based FPGAs using the FLIPPER fault injection platform. 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT07), 2007

[10] Sterpone L, Violante M. A new analytical approach to estimate the effects of SEUs in TMR architectures implemented through

SRAM-based FPGAs. IEEE Trans Nucl Sci, 2005, 52(6): 2217

[11] Sterpone L, Violante M, Sorensen R, et al. Experimental validation of a tool for predicting the effects of soft errors in SRAM-based FPGAs. IEEE Trans Nucl Sci, 2007, 54(6): 2576

[12] Alderighi M, Casini F, Angelo S, et al. Soft errors in SRAM-FPGAs: a comparison of two complementary approaches. IEEE Trans Nucl Sci, 2008, 55(5): 2267

[13] Quinn H, Graham P, Pratt B. An automated approach to estimating hardness assurance issues in triple-modular redundancy circuits in Xilinx FPGAs. IEEE Trans Nucl Sci, 2008, 55(4): 3070

[14] Xilinx Inc. Virtex FPGA series configuration and readback, Xilinx application note XAPP138, Version 2.8, 2006. http://www.xilinx.com/support/documentation/application notes

[15] Xilinx Inc., Virtex Series Configuration Architecture User Guide, Xilinx Application Note XAPP151, Version 2.8, 2004. http://www.xilinx.com/support/documentation/application notes

[16] Xilinx Inc. The JBits 2.8 SDK for Virtex, 2001. http://www.xilinx.com/labs/projects/jbits

[17] Filho C, Kastensmidt F, Carro L. Improving reliability of SRAM-based FPGAs by inserting redundant routing. IEEE Trans Nucl Sci, 2006, 53(4): 2060

[18] Filho C, Kastensmidt F, Carro L. Improving reliability of SRAM-based FPGAs by inserting redundant routing. Military and Aerospace Applications of Programmable Logic Devices (MAPLD), Las Vegas, Nevada, USA, 2005

[19] Graham P, Caffrey M, Zimmerman J, et al. Consequences and categories of SRAM FPGA configuration SEUs. Military and Aerospace Programmable Logic Devices International Conference(MAPLD), Washington DC, 2003

[20] Ceschia M, Violante M, Reorda M, et al. Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs. IEEE Trans Nucl Sci, 2003, 50(6): 2088

[21] Asadi G, Tahoori M. An analytical approach for soft error rate estimation of SRAM-based FPGAs. Military and Aerospace Applications of Programmable Logic Devices (MAPLD), Washington, D.C., 2004