Experimental emulation of FPGA bitstream faults in combinatorial circuits

Jiří Kvasnička, Pavel Kubalík, Hana Kubátová

Department of Computer Science and Engineering
Czech Technical University in Prague
Karlovo nam. 13, 121 35 Prague 2
{kvasnj1, xkubalik, kubatova}@fel.cvut.cz

Abstract. Abstract—The main aim of our research is to design dependable circuit in FPGA. To make a real dependability model the real effects of injected errors and faults have to be studied. We proposed a hardware fault emulator here. The emulator deals with single-bit change in bitstream. Emulation is performed in user-selected area. Look-Up-Tables, cell interconnection, cell-to-bus connection and routing resources are considered. Other FPGA resources are not considered. Only combinatorial circuits and benchmarks were measured due to our knowledge of FPGA resource limitation. All tests were performed on Atmel FPSLIC architecture.

1 Introduction

The size, versatility, and price of commercial Field Programmable Logic Arrays (FPGAs) allows replacing ASICs in many applications [1]-[3]. The SRAM, which the FPGA configuration is based on, is sensitive on Single Event Upset SEU [4]. The SEU can be caused by high-energy particles hitting the silicon and changing the logic state of the SRAM cell.

The main aim of our research is to design dependable circuit in FPGA. To make a real dependability model the real effects of injected errors and faults have to be studied. This paper is about the detailed analysis about the possible faults and their effects to the final design implemented in FPGA. Experiments apply fault injection into the bitstream and emulation method [5-8].

The SEU impact to FPGA was also performed by another research group [6]. Here the possible situations are divided into 6 groups. Models and experiments were done for ATMEL FPGAs. Results could lead to reliability parameters increasing in our future research. Some publications have focused on reliable systems based on a single FPGA using a Triple Module Redundancy (TMR) structure inside [9] and [10]. These techniques can be based on knowledge of FPGA reliability resources cost.

The structure of the paper is following: The section 2 describes our proposed classes for Area in FPGA and possible classes of faults in FPGA structure. The section 3 describes our emulator, where the measurement was performed. The section 4 shows result from the experimental fault injection, and the section 5 concludes the results and the contribution of the measurement.

2 FPGA fault analysis

The main problem in detailed studies of real faults and their effects to the function of design implemented in FPGA is the very confined knowledge of front-end of FPGA professional design tools (due to licensees). Here the detailed experiments and methods how to get know this information are presented.

2.1 FPGA resources

The FPGA resources were divided into disjoint sets (listed below), which are specified by their location and function.

1. ***LUT***: It holds the logic function in SRAM memory.
2. ***Cell interconnection***: The configuration of Logic cells; these bits are responsible for the LUT correct inputs selection; feedback in logic cell, correct output selection (registered/nonregistered function, 3- or 4-input LUT organization).
3. ***BUS to Cell/Cell to BUS***: This is a bidirectional connection, which connects the Logic Cell to one of BUS plane.
4. ***BUS crossing***: This is a connection in the centre of perpendicular bus crossing, which allows connections between these lines.
5. ***BUS repeater***: This is a simple 4-port switchbox. It allows driving of each wire from every input.
6. ***Forbidden***: These are bits, that have their own place in bitstream, but a physical SRAM cell is probably not assigned to them. This is caused by the bitstream byte organization. It is also possible that these bits have other (for us unknown) meaning. We assume, that these bits does not exist therefore should not be tested.
7. ***Unexplored***: other resources, which were not listed above. The position in the bitstream implies their usage, although their exact role in the bitstream has not been completely analysed. Parts of *RAM*, *reset* and *clock* resources are expected to lie in this category.

This list is not complete. ***RAM***, ***reset***, ***clocks*** and ***I/O pad*** (only partially covered here by *unexplored* set) are missing in the list. Our emulator almost doesn't use them.

However incompleteness of the list does not imply that SEU occurrence in nonlisted region could not destroy the design. The time needed to determine the category, which the bit belong in, corresponds to $O(1)$ time complexity.

2.2 Fault groups

Primary goal of fault division is a separation of bits, which can never influence on the function of the loaded design, and bits which can lead to the modification of the design. Such a classification is possible only when the complete structure of FPGA is known (and corresponding bitstream position is known).

A class cannot be correctly evaluated without knowledge of wire state (Function, constant or High-Z) and current bit value. The computation is therefore based on the design bitstream, because the distribution into these groups is specified by the physical layout of the FPGA.

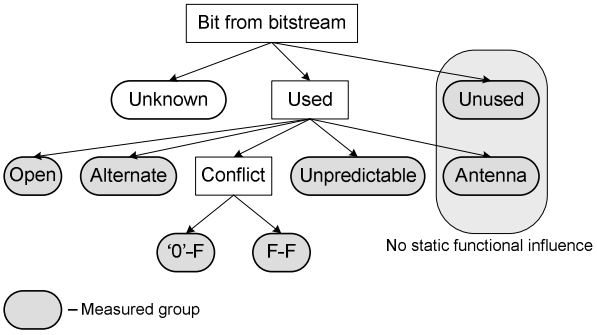

Fig. 1. Fault groups

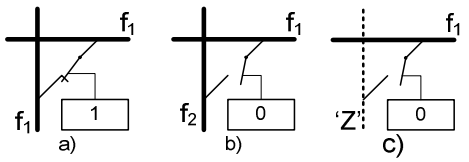

Fig. 2. Used bits: a) connection (open possible), b) no connection (short circuit possible) and c) no connection (antenna possible)

Every fault belongs to one fault group, as shown in Fig. 1. In the first approximation, the bit is placed in *Used*, *Unused* or *Unknown* group.

1. ***Unused***: this bit does not concern the design area. No signal path can exist between the design and wires associated with the fault. Therefore neither static nor dynamic design changes in the design are expected. Although, such a fault could potentially lead to higher static current consumption when '0' collides with '1'.
2. ***Used***: This group belongs to bits, which are primary created by the design of connecting resources (a sample shown on *Bus crossing* in Fig. 2.a or by standing in the critical position, where switching this resource can lead to *conflict* (Fig. 2.b) or *alternation*.
   Any bit from this group influences an active part of the design. Alternation of this bit can lead to the design modification, shorts in design, open on the datapath, which can functionally alter the design. A special group, *antenna*, can alter only the dynamic behavior by adding an extra capacitance to load (Fig. 2.c).
3. ***Unknown***: A class of bits, whose correct class can not be evaluated. This, in simplification, can be caused by unknown state on the wire, or by missing information about the bit meaning in FPGA resources.

2.2.1 Subcategories of used bits

A group of used bits covers many versatile types of fault. Almost all of them can lead to design functional alternation. The only exception is Antenna group.

All the Figures in this subsection show both the original (correct) configuration and the modified configuration (after the fault).

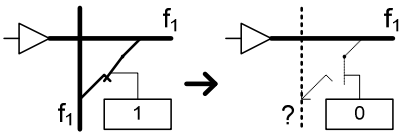

Fig. 3. Open at the bus crossing

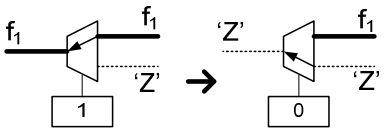

Fig. 4. Open in 2:1 mux

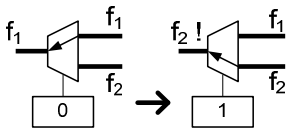

Fig. 5. Alternate at 2:1 multiplexor

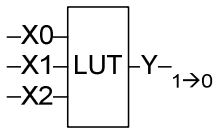

Fig. 6. Alternate at LUT

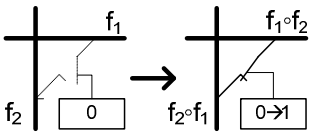

Fig. 7. Conflict at the bus crossing

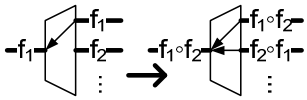

Fig. 8. Conflict in multiplexor – multiple input

1. ***Open***: The basic model of this class is a wire interruption, which can have many different origins in the FPGA architecture. The most lucid case is an open in bus crossing (Fig. 3).
   Another case of open is in 2:1 mux, where the resulting function (on the left) suddenly has no source driver on the fault occurrence. This case is shown on Fig. 4 and assumes transfer-gate mux realization:
2. ***Alternate***: These bits alternate the design without any conflict on the bus. A 2:1 mux with configuration shown in Fig. 5 is a typical example of alternate group. In some situation (e.g. Fig. 5), the *alternate* fault can be viewed as a special combination of *open* and *antenna*, which does not leave any wire undriven and which does not creates conflict when connecting another signal.

   A special case, when both f_1 and f_2 are the same function, is possible, but these cases are neither detected nor separated in our fault emulator. Another example is a bit alternation in truth table of LUT. (Fig. 6).
3. ***Conflict***: This is a special category defined by connecting of two or more driven wires. This conflict leads to a short circuit between power supply and ground through the drivers. The result of conflict is hard to predict, unless a detailed FPGA layout is known. A simple conflict can occur on BUS crossings (Fig. 7). The "$\circ$" operator in Figures 7 and following should be interpreted as dominance: if one of the operands can drain more current, that will be result of the "$\circ$" operator.

   Conflicts can also occur in multiplexers when selecting more than 1 input (see Fig. 8). Another possibility of signal conflict is in bus network. In altered state, a conflict on a common bus wire can occur (Fig. 9), with the same current drain strength of the drivers.

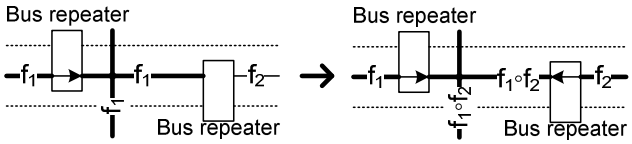

Fig. 9. Conflict at the bus repeater

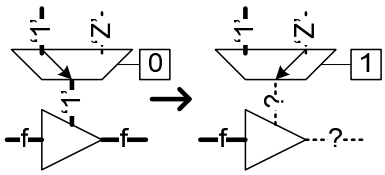

Fig. 10. Unpredictable output enable buffer

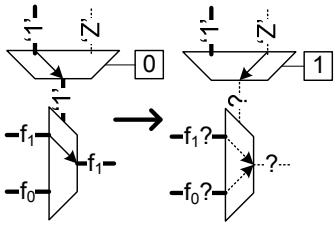

Fig. 11. Unpredictable mux selector driver

A conflict is separated into 2 subcategories:

- "0-F" in cases, where any function conflicts with constant '0'
- "F-F", where conflict is between two non-constant functions.

4. ***Unpredictable*** faults are a special case of open, where the default logical value '1' is substituted with 'Z'. Two cases in FPSLIC architecture are presented at Figures 10 and 11. These *unpredictable* faults were separated from other faults only because of unknown physical layout of these elements.
5. ***Antenna***: A special case of a used bit, where an unused wire is connected to the datapath. This fault statically has no influence on the design function. An example of antenna fault is shown in bus crossing at Fig. 12. When the appended wire has not assigned a driver, the appended line is driven from the first input (Fig.13.)

Another multiplexor configuration is shown in Fig. 14. The multiplexor has no selected input. The output wire of the multiplexor is not driven nor read. Selecting the input will not harm the design, it append an unused wire to the datapath.

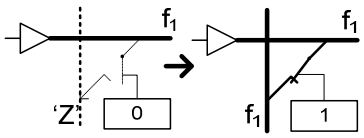

Fig. 12. Antenna at the bus crossing

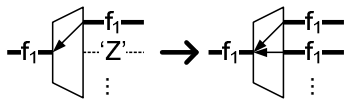

Fig. 13. Antenna in the mux, multiple inputs

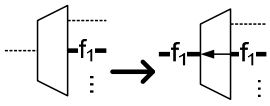

Fig. 14. Output antenna at mux with no input selection

Since a bus repeater has a build-in driver, the appended output should be not as big capacitance as other listed cases. Despite, such case is also considered as antenna.

3 Our Emulator

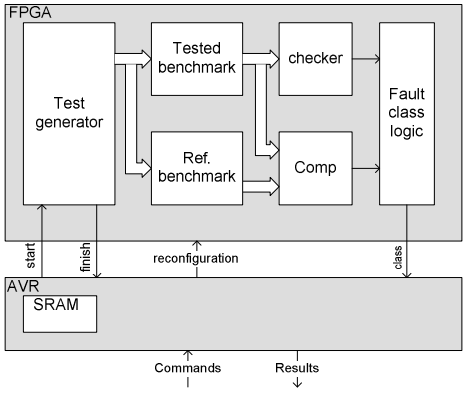

Fig. 15. The FPGA fault emulator overview

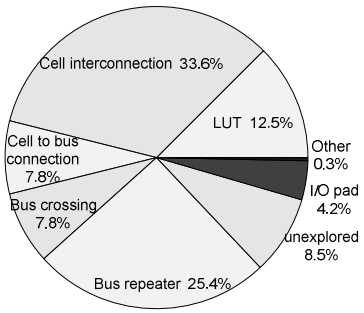

Fig. 16. Resources in Bitstream distribution for Atmel At40 FPGA. Dark slices (IO pads and Others) were not tested

We used a hardware emulator described in [8], which was extended by ability to test bits listed in chapter 2.1. The SEU fault is emulated in the FPGA through the dynamic reconfiguration. Each single bit in the bitstream represents one possible fault, which is injected into the FPGA by writing the inverted value into the design.

The FPGA is virtually divided into two areas: *fault injection* area and *fault safe* area. The emulator (Fig. 15) was implemented in the ATMEL FPSLIC device.

In this paper, the resulting class of the fault is interpreted only as a "fault is modifying the design" or "fault is not modifying the design." However, more resulting categories according to [11] is also possible to obtain from the emulator.

4 Results

The first important result is the number of bits in every FPGA *resource* and every *fault class*.. The S1488 benchmark is presented in Table 1. Each column represents one resource from section 2.1. Each row represents one fault from section 2.2.1. Total number of bits for each resource and fault category can be found in crossing of related column and row. Impossible combinations of categories are marked by "–"symbol.

Table 1. Total bit count in s1488 benchmark

	LUT	Cell interc.	Bus/ cell	Bus cross.	Bus repeat.	Forbid.	Unexp_.
Unused	13 304	23 286	6 155	6 434	18 916	1 104	–
Alternate	4 360	3 417	–	–	–	–	–
Open	–	730	1 643	363	1 077	–	–
Conflict 0F	–	2 228	–	–	–	–	–
Conflict FF	–	5 133	817	1 181	9 403	–	–
Antenna	–	12 108	2 425	3 062	5 824	–	–
Unpredictable	–	570	–	–	–	–	–
Unknown	–	–	–	–	–	–	11 740

Table 2. Ratio of bits altering the design to all bits in category

	LUT [%]	Cell interc. [%]	Bus to cell [%]	Bus cros-sing [%]	Bus repea-ter [%]	Forbidden [%]	Unexplore d [%]
Unused	0.0	0.0	0.0	0.0	0.0	0.0	–
Alternate	84.8	79.0	–	–	–	–	–
Open	–	98.8	97.6	100.0	100.0	–	–
Conflict_0F	–	87.9	–	–	–	–	–
Conflict_FF	–	64.9	59.6	38.6	89.1	–	–
Antenna	–	0.0	0.0	0.0	0.0	–	–
Unpredictable	–	19.6	–	–	–	–	–
Unknown	–	–	–	–	–	–	3.3

Interpretation of Table 1: Although the place and route tool from Atmel reports 329 used logic cells, more logic cells are occupied. Additional logic cells are used on routing. However, less LUT bit number (only 4360 bits) is in *alternate* fault group instead of more than 5280 bits expected. This less number is caused by utilization of LUT, which is not always used as a 4-input LUTs or two 3-input LUTs.

Table 2 shows the ratio between bits, which modify the design during the fault injection, and all bits in corresponding *area* and *fault category*. The layout of the results in this table is similar to Table 1.

The assumption, that the *antenna* and *unused* bits have no influence on the design function, was confirmed by these experiments. *Open* fault have significantly higher probability that can change the design. On the other hand, a *conflict* between two functions at *bus crossings*, *cell interconnections* and *bus/cell connection* have unexpectedly low ratio of altering bits to all bits. The reason of this result has not yet been fully analyzed. It may indicate a higher probability of conflict between same functions in the logic cell or significantly different strengths of drivers in logic cells.

Table 3 shows several benchmarks and their ratio of bits, which can change the design, to all bits in corresponding *fault* category. The first line in the Table 2 shows a total LUT bit number in alternate category as an indicator of the benchmark size.

Table 3. Ratio of altering bits in fault categories for different benchmarks

	5xp1	alu1	alu2	alu3	b11	b12	br1	bw	s1488	s1494
used LUT bits	388	676	1102	1090	420	650	822	834	4360	4042
Unused	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Alternate	90.0	92.3	84.7	84.5	83.6	86.0	81.0	80.5	82.3	81.0
Open	97.6	99.3	98.1	97.8	98.4	98.2	99.6	98.9	98.7	98.9
Conflict 0F	93.6	94.4	87.2	87.4	89.2	86.1	87.7	88.1	87.9	88.1
Conflict FF	84.7	86.0	80.6	79.5	79.9	82.2	77.7	81.5	76.5	76.4
Antenna	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Unpredict.	24.7	19.8	10.9	15.3	25.8	48.9	11.1	20.4	19.6	22.6
Unknown	0.2	0.3	0.6	0.5	0.2	0.3	0.5	0.4	3.3	3.0

5 Conclusion

We are able to obtain precisely how many bits can change the design, which is actually mapped and running into the FPGA. This is a significant pre-requirement in dependability modeling and calculations. Moreover, we are able to separate bits, which can not change the design from the whole bitstream.

The obtained results opens a new field of application in adjusting the synthesis and place&route tools to compile a design, which would be slightly more resistant to SEU - but with noticeable worse delays in FPGA and the maximum frequency decrease.

Acknowledgements

This research has been partially supported by MSMT under research program MSM6840770014 and FI-IM4/149 grant.

References

1 Dobiáš, R., Kubátova, H.:“FPGA Based Design of Raiway's Interlocking Equipment”, In Proceedings of EUROMICRO Symposium on Digital System Design. Piscataway: IEEE, 2004, pp 467-473.
2 Ratter, D.:“FPGAs on Mars”, www.xilinx.com, Xcell Journal Online, 2004.
3 Actel Corporation.: “Historic Phoenix Mars Mission Flies Actel RTAX-S Devices”, www.actel.com, 2007.
4 Normand, E.:“Single Event Upset at Ground Level,” IEEE Transactions on Nuclear Science, vol. 43, 1996, pp. 2742-2750.
5 Kafka, L., Novak, O.: “FPGA-based fault simulator”, In Proceedings of the 2006 IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems DDECS2006, CTU Prague 2006, vol. 1, pp. 274-278.
6 Bellato, M., Bernardi, P., Bortalato, D., Candelaro, A., Ceschia, M., Paccagnella, A., Rebaudego, M., Sonza Reorda, M., Violante, M., Zambolin, P.: “Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA.” Design Automation Event for Electronic System in Europe 2004, pp. 584-589.
7 Graham, P., Caffrey, M., Zimmerman, J., Sundararajan, P., Johnson, E., Patterson, C.:"Consequences and Categories of SRAM FPGA Configuration SEUs", MAPLD International Conference, Washington DC, 2003, Paper C6.
8 Kubalík, P., Kvasnička, J., Kubátová, H.:“ Fault Injection and Simulation for Fault Tolerant Reconfigurable Duplex System”, In Proceedings of DDECS 2007, pp. 357-360
9 Sterpone, L., Violante, M.: “A design flow for protecting FPGA-based systems against single event upsets “, DFT2005, 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 436 – 444.
10 Berg, M.: “Fault Tolerance Implementation within SRAM Based FPGA Design Based upon the Increased Level of Single Event Upset Susceptibility”, In Proceedings of the 12th IEEE International On-Line Testing Symposium, IOLTS'06, pp. 89-91, July 2006.
11 Kafka L., Kubalík P., Kubátová H., Novák O.: “Fault Classification for Self-checking Circuits Implemented in FPGA”, Proceedings of IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop. Sopron University of Western Hungary, 2005, pp. 228-231