FLINT: Layout-Oriented FPGA-Based Methodology for Fault Tolerant ASIC Design

Rochus Nowosielski, Lukas Gerlach, Stephan Bieband, Guillermo Payá-Vayá, and Holger Blume
Institute of Microelectronic Systems, Leibniz Universität Hannover, Appelstraße 4, 30167 Hannover, Germany
Email: {nowosielski,gerlach,bieband,guipava,blume}@ims.uni-hannover.de

***Abstract*—Research of efficient fault tolerance techniques for digital systems requires insight into the fault propagation mechanism inside the ASIC design. Radiation, high temperature, or charge sharing effects in ultra-deep submicron technologies influence fault generation and propagation dependent on die location. The proposed methodology links efficient fault injection to fault propagation in the floorplan view of a standard cell ASIC. This is achieved by instrumentation of the gate netlist after place&route, emulation in an FPGA system and experiment control via interactive user interface. Further, automated fault injection campaigns allow exhaustive fault tolerance evaluations taking single faults as well as adjacent cell faults into account. The proposed methodology can be used to identify vulnerable cell nodes in the design and allow the classification of placement strategies of fault tolerant ASIC designs.**

I. Introduction

Complex system-on-chip implementations are also used in safety critical applications and under harsh environmental conditions. These chips are composed of millions of transistors. The failure of only one of these transistors can cause the whole system to malfunction. This is prevented by fault tolerant design techniques on system level, chip level - subdivided into architecture and circuit level, and technology level. The proposed methodology addresses fault tolerant design at architecture level.

At this level, proposals for fault tolerant design techniques already exist but novel techniques are still required in order to cope with the challenges. This methodology targets faults from radiation, high temperature and parasitic effects in ultra-deep submicron technologies:

Energetic radiation particles impact the active region of the semiconductor and generate electron-hole pairs along their path. Radiation related faults, also known as single event effects (SEE) are induced from ionization processes in the semiconductor material. That ionization process causes faults in digital logic, that can lead to failures and system malfunction. Any kind of radiation: alpha particles, accelerated neutrons, gamma rays, heavy ions, or electro-magnetic interference has this effect. The intensity and nature of radiation is highly dependent on the environment: like satellites, artificial radiation at ground level (e.g. inside nuclear power plants), or moderate natural radiation in underground installations like water pumps for geothermal applications.

In high temperature environments electronics are exposed to very high thermal stress over their lifetime when operated at junction temperatures above $250\,^{\circ}\mathrm{C}$. These high operation temperatures reduce the device lifetime significantly. Due to wearout, reliability issues arise for electronic components at end-of-life stage. Wearout related faults can be described as SEEs, i.e. degradation of semiconductor devices decreases the timing slack and leads to setup violations along digital logic paths. Electromigration causes an increased wiring resistance and even opens inside the circuit.

Faults from parasitic effects in *ultra-deep submicron technologies* become relevant for ASIC technologies at $28\,\mathrm{nm}$ feature size and below: single event induced charge sharing faults in deep submicron technologies manifest themselves in a similar fashion to SEEs [1]. Since, charge sharing couplings are related to small distances between the devices and also small capacities per device [2], they emerge with the scaling progress of semiconductor devices.

These different application fields and fault sources require customized fault tolerant digital architectures. The FLINT (**F**au**L**t **IN**jection **T**ool) methodology presented in this paper aids an ASIC designer in the evaluation of well-established fault tolerance techniques. It also assists in the research of challenges addressed in this introduction. This is achieved by means of novel fault injection techniques in contrast to other proposals [1], [3]–[6]: Interactive fault injection under consideration of adjacent cells and fault propagation monitoring, both in floorplan view.

The paper is organized as follows: Section II provides details about the proposed methodology. In Section III supported modes for fault tolerance research of digital ASIC designs are presented based on evaluation experiments with an 8-bit processor for ambient temperatures above $250\,^{\circ}\mathrm{C}$. Section IV gives a conclusion of the presented work.

II. Layout-oriented Fault Injection Methodology

The proposed methodology addresses the ASIC architect's perspective on the digital design: Besides the overall fault tolerance of the complete system or a specific design block, the research of effective fault detection and recovery strategies makes it important to understand, how the fault is propagated in a complex digital system. For this task, four key aspects can be identified that are considered in this work:

- placement dependent fault tolerance evaluation
- fine-grained control on the injected faults
- visualization of fault propagation in the floorplan view
- computation of SER with in considerable time

978-3-9815370-4-8/DATE15/©2015 EDAA

A. Placement-Driven Fault Injection

Visualization of fault propagation in floorplan view helps significantly the understanding of the topological dependencies inside the ASIC. The proposed methodology supports placement dependent fault tolerant design evaluation by providing two features: The first one is *layout oriented design data input*. The gate level description of a standard cell design after place&route is the topmost unique description of the future ASIC in terms of abstraction levels. The data to process is placement location of the standard cells stored in Cadence DEF format. This corresponds to the floorplan view. Signal interconnections are stored in Verilog netlist format.

The second feature is *fine-grained fault injection*. Transient and permanent faults occur in a particular location on chip. Fault injection in floorplan view allows selecting gates that, due to their placement, can be affected by one fault, contemporary. Inside standard cells, faults propagate towards gate outputs. Thus, fault injection at standard cell outputs, models this faulty behavior. The amount of injected faults in the floorplan view is arbitrary. Each fault can be specified by X,Y coordinates and radius.

Although, place&route is highly technology dependent, the proposed approach is not: the fault injection additions required are performed on the gate outputs, only. These are automatically identified in the netlist during the instrumentation process.

B. Gate-Level Instrumentation

In order to inject faults at each gate output, gate instrumentation is proposed. For this, each gate is replaced by a corresponding *instrumentation unit*. The instrumentation of combinatorial gates is a simplified variant of sequential gate instrumentation. Thus, the combinatorial gate instrumentation is explained first but is also valid for sequential gates. Figure 1 depicts the instrumentation unit of an AND2 combinatorial gate. In this unit, the original output of the gate is connected to the *Fault MUX* labeled `MUX_12`. It controls the fault manifestation at the instrumentation unit's output Q and selects between the error free case, bit-flip, stuck-at-1, and stuck-at-0. The fault selection is done by the signals `seq_in_1` and `seq_in_2`. These signals are outputs of the `DFF_1` and `DFF_2` flip-flops in the driving, i.e. previous, instrumentation unit. The internal gate output can be read out via `MUX_3` and `DFF_3`. All D-flip-flops with the same index are chained across all instrumentation units. Thus, three scan chains are created in order to reduce scan time.

Figure 2 depicts the instrumentation unit for a sequential D-flip-flop. Two additions can be observed: First, the original flip-flop `DFF_0` belongs to a different clock domain than the configuration flip-flops `DFF_1`, `DFF_2`, and `DFF_3`. That way, the scan chains can be read out at a faster clock speed. Second, the multiplexer `MUX_0` is added. This multiplexer is controlled by the `write` signal and selects between original gate input, bit-flipped value, old value and load value. The last option allows the restoration of a particular design state and to perform different analysis runs.

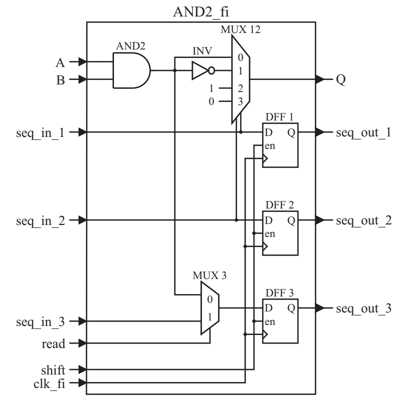

Fig. 1. Instrumentation unit for a combinatorial AND2 standard cell.

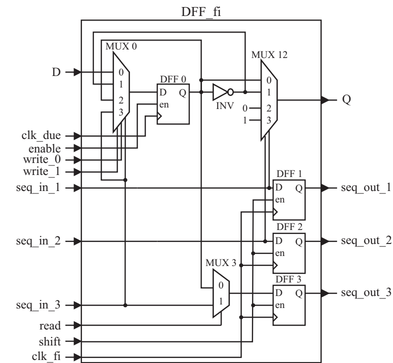

Fig. 2. Instrumentation unit for a sequential D-flip-flop standard cell.

C. Work Flow for Fault Tolerance Analysis

The proposed work flow is twofold and shown in Figure 3: In the first step, a standard ASIC design flow is performed. After synthesis, the place&route tool creates a Verilog netlist and a DEF file with place&route data of the designed ASIC. This unique ASIC description is direct input for the second step. For instrumentation, the netlist is modified: the instantiated standard cells are replaced by the corresponding instrumentation units. Further, the fault injection scan chains are connected. Because of the explicit scan chain design, the FPGA synthesis tool performs no logical optimizations on the netlist. This assures a unique correspondence between emulated gates and gates placed in the floorplan view.

For interactive floorplan view generation, placement information from the DEF file and geometrical information from the technology library is merged in the graphical fault analysis tool. This tool communicates with the emulation system, which can be any convenient FPGA development environment. The proposed methodology has been proofed on a Xilinx ML605 evaluation board in conjunction with the UEMU framework [7]. This framework is interfaced by a control module, which arbitrates between the on chip bus inside the UEMU framework and the evaluated design. Host communication with the UEMU system uses raw gigabit Ethernet. Once, the FPGA bitstream is transferred to the FPGA, all fault injection runs are controlled by the graphical fault analysis tool, which was implemented in Java.

D. Interactive User Interface

The presented FLINT framework can be used fully automated. For this, the UEMU framework provides a C language interface. However, for interactive fault injection and propagation monitoring, a convenient GUI implementation in Java is shown in Figure 4. The GUI window is composed of three panes. The center pane is the floorplan view showing the standard cell placement generated during ASIC place&route. Due to hierarchical tool flow, each standard cell can be assigned to the corresponding HDL module of the design. This module correspondence is used for colored highlighting. Since each cell is represented by a button widget, mouse selection events are used in order to display cell information or to apply a fault model to a cell or adjacent cells in radius-based selection mode.

After fault setup, a corresponding configuration stream is generated and transferred to the UEMU system. The communication protocol is displayed in the right pane. The scan feature introduced during the instrumentation process is used to read out the logical state of the complete design after each clock cycle and to load arbitrary flip-flop states into the design. First, a fault free processing sequence is captured. Then the initial circuit state is restored, the fault configuration is applied, and the faulty processing sequence is captured again. In that way, information is collected in order to mark gates with faulty logical state in the graphical interface.

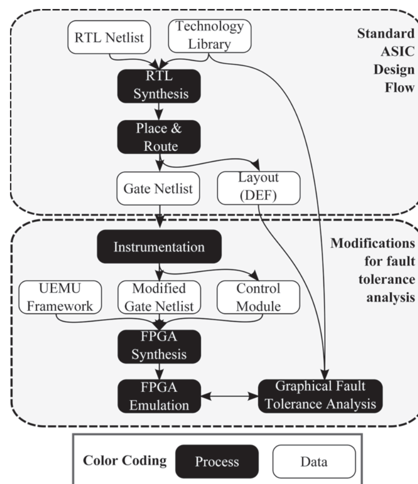

Fig. 3. Work flow of proposed fault tolerance design methodology

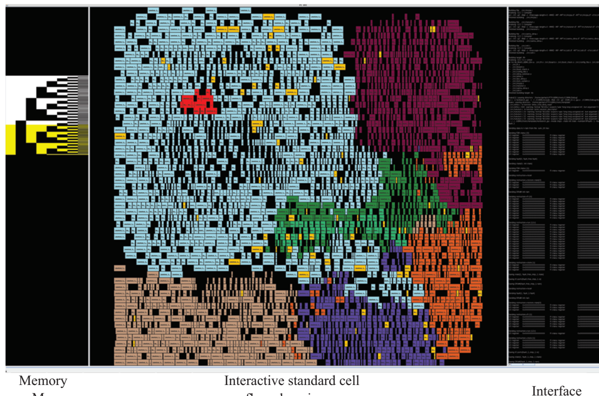

Fig. 4. Visualization of failure setup and propagation for an 8-bit RISC processor core.

TABLE I
SER ANALYSIS FOR HC11 PROCESSOR DURING BOOT UP SEQUENCE.

	silent	latent	RAM latent	failure
Baseline	78.51%	5.73%	14.00%	1.76%
Hamming encoded RF	83.51%	5.05%	9.89%	1.54%

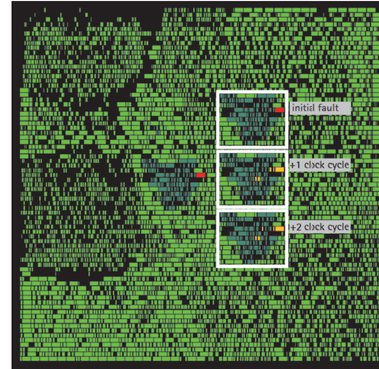

Fig. 5. Visualization of fault propagation inside a design module over two clock cycles in interactive mode.

Figure 4 shows a snapshot from an exemplary run. Red colored cells indicate activated faults in the register file. Yellow colored cells display resulting failures along the path from fault origin to the next flip-flop. A slider at the bottom of the GUI window allows a selection of the captured observation points. In this way, the cycle-true fault propagation can be replayed in the floorplan view. Each observation point is also a check point that can be loaded back into the design. This way, resuming evaluation with different fault settings is realized.

In the left pane the appropriate memory content is displayed. Faults and failures are marked in different colors. For larger memories a clustered view is possible. Please note: Memory cells in the floorplan view can be integrated from DEF file, also. In the presented GUI, this was skipped in order to put the focus solely on architecture research. Instead, faults in memory cells are injected through direct memory access in the UEMU framework.

III. EXEMPLARY RESEARCH CASES

The presented fault tolerance methodology and framework supports different research scenarios as follows:

Soft Error Rate (SER) with fault classification is an automated task of fault injection and classification of resulting failures. FLINT makes SER analysis possible and supports the fault classification proposed in [8]. Exemplary results from an analysis of an HC11 processor core during startup sequence are shown in Table I. Here, the reduction of failure rate due to implementation of Hamming Codes in the register file (RF) can be observed.

Interactive visualization of fault propagation is intended for detailed research of the implemented fault tolerance mechanisms in a given scenario. In that mode, the designer can

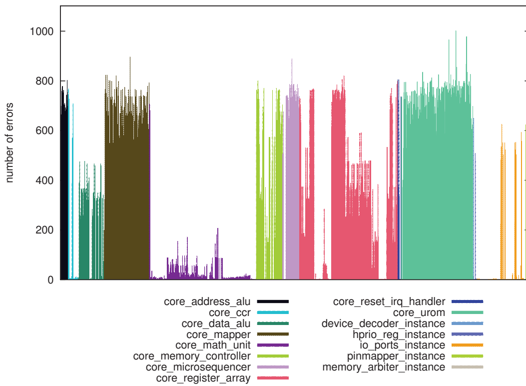

Fig. 6. Failing gate level nodes for exhaustive fault injection with one fault per node. Nodes are sorted by module hierarchy.

Fig. 7. Heatmap of design vulnerability due to adjacent placement.

stimulate a single cell or a region of cells and observe, cycle true, the manifestation of that fault inside the design. Figure 5 depicts a sequence of such interactive fault propagation.

Exhaustive fault stimulation campaigns at gate level allow the designer to evaluate the potential vulnerability of each digital cell. For that measurement, one fault per execution step, e.g. clock cycle, is injected and the amount of failing cells is counted. This has to be repeated for all cells inside the netlist. The accumulated amount of faults induced by a particular faulty cell is the measure of vulnerability of that particular cell. Figure 6 shows an exemplary result for an HC11 processor core. Here, the vulnerability of each cell inside the design is grouped by the corresponding module of the processor core.

In *placement dependent fault analysis* mode, the designer can evaluate the placement impact on the design vulnerability. This evaluation is related to the exhaustive fault stimulation described before. Now, two adjacent cells are selected to be faulty in each evaluation step. Again, this is done exhaustively for all cells. An exemplary result is shown in Figure 7. The color intensity corresponds to the vulnerability of the particular location in the die area.

As indicated in Table II, the time taken for the exhaustive fault injection campaigns is 20 and 46 seconds, respectively. For comparison, simulation times at RT-level and gate level are provided as well. Please note: These times are valid for plain simulation of the design without any further modifications for fault injection. Even in this comparison the presented FLINT implementation is more than 25 times faster than

TABLE II
FLINT EMULATION TIME COMPARED TO TIME REQUIRED FOR AN RTL AND GATE LEVEL SIMULATION IN MODELSIM WITHOUT FAULT INJECTION (FI).

	FLINT emulation w/ FI (sec)	RTL simulation w/o FI (sec)	gate level simulation w/o FI (sec)
single fault	20	538	4313
adjacent faults	46	1208	9696

the RTL simulation and more than 200 times faster than the corresponding gate netlist simulation.

IV. Conclusion

FLINT provides a novel, technology independent approach for interactive as well as automated fault injection at floorplan level. The proposed work flow allows an interactive approach to understand failure propagation inside the circuit at an early ASIC design stage. It aids the hardware architect in evaluation of fault tolerance techniques inside a given design. This interactive aspect becomes possible due to fast emulation-based analysis. Compared to an equivalent gate level simulation, this methodology is more than 200 times faster. This speed gain also makes exhaustive analysis techniques possible. The use cases presented an exhaustive campaign with one fault and a placement oriented exhaustive campaign with two adjacent faults. In ongoing work, this methodology will be used in order to identify vulnerable gate nodes inside the design and in order to identify a placement strategy that exposes minimal fault sensitivity. Based on these results, a hardening of digital high temperature ASIC components will be performed.

References

[1] L. Entrena, A. Lindoso, E. S. Millan, S. Pagliarini, F. Almeida, and F. Kastensmidt, “Constrained placement methodology for reducing SER under single-event-induced charge sharing effects,” *Nuclear Science, IEEE Transactions on*, vol. 59, no. 4, pp. 811–817, 2012.

[2] B. D. Olson, D. R. Ball, K. M. Warren, L. W. Massengill, N. F. Haddad, S. E. Doyle, and D. McMorrow, “Simultaneous single event charge sharing and parasitic bipolar conduction in a highly-scaled SRAM design,” *Nuclear Science, IEEE Transactions on*, vol. 52, no. 6, pp. 2132–2136, 2005.

[3] H. Guzman-Miranda, J. Tombs, and M. Aguirre, “FT-UNSHADES-up: A platform for the analysis and optimal hardening of embedded systems in radiation environments,” in *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*. IEEE, 2008, pp. 2276–2281.

[4] H. Zheng, L. Fan, and S. Yue, “FITVS: A FPGA-based emulation tool for high-efficiency hardness evaluation,” in *Parallel and Distributed Processing with Applications, 2008. ISPA’08. International Symposium on*. IEEE, 2008, pp. 525–531.

[5] L. Entrena, M. Garcia-Valderas, R. Fernandez-Cardenal, A. Lindoso, M. Portela, and C. Lopez-Ongil, “Soft error sensitivity evaluation of microprocessors by multilevel emulation-based fault injection,” *Computers, IEEE Transactions on*, vol. 61, no. 3, pp. 313–322, 2012.

[6] W. Mansour, R. Velazco, and G. Hubert, “Error-rate estimation combining SEE static cross-section predictions and fault-injections performed on HDL-based designs,” *Nuclear Science, IEEE Transactions on*, vol. 60, no. 6, pp. 4238–4242, 2013.

[7] M. Kock, S. Hesselbarth, M. Pfitzner, and H. Blume, “Hardware-accelerated design space exploration framework for communication systems,” *Analog Integrated Circuits and Signal Processing*, pp. 1–15, 2013.

[8] L. Entrena, A. Lindoso, M. G. Valderas, M. Portela, and C. L. Ongil, “Analysis of SET effects in a PIC microprocessor for selective hardening,” *Nuclear Science, IEEE Transactions on*, vol. 58, no. 3, pp. 1078–1085, 2011.