Reliability Bottlenecks in Integrated Parallel Fault-Tolerant Systems

Bernhard Fechner, Department of Mathematics and Computer Science, Parallel Computing and VLSI Group, FernUniversität in Hagen, 58084 Hagen, Germany

Abstract

The appearance of multithreaded, multi- and manycore systems has led to another advance in performance. Such systems are denoted as integrated, as long as there are electrical dependencies between functional units, i.e. multiple cores integrated on a die. With the appearance of such integrated systems, several questions concerning fault propagation arose. First, if one component fails, how likely is a faulty behavior of other components, how likely is the fault going to propagate between components? Second, what is the overall reliability of such a system? It is important to answer these questions prior to implementation, because the total costs of the final product shall be as small as possible. There are numerous fault models, especially on the physical level, dealing with propagation and development of electrical current. Computation time is essential when considering fault simulation of large systems. Our approach combines different abstraction levels in one fault model, allowing the generalized modeling of faults. Hence every fault can be modeled if its effect can be defined by an analytical function. The first level is the physical level, covering the physical effects of a fault, second a component and routing model and the last the behavioral modeling of components by finite state machines. The model can cover the whole range of parallel devices. It can help to improve reliability of current and future parallel fault-tolerant systems by identifying the underlying bottlenecks. The function of the model is exemplarily shown by applying it to an FPGA, identifying switchboxes as the main reliability bottleneck.

1 Introduction

The nature of modern integrated parallel systems such as multi- or manycore systems does not only offer the potential to increase computing performance but also increase reliability. NoCs or SoCs are examples of integrated parallel systems. They can help to increase the overall reliability of an n-modular redundant system, but only if the reliability of a single component is increased. Usually it is not known prior to an implementation which components have to be harnessed against faults and how high the MTTF is. Therefore, a model has to be developed. One of the biggest problems when constructing a fault model is to validate it. The second issue is abstraction and exactness. While an abstract model can be quickly simulated and implemented, its precision will suffer. If the model is too complex, simulation time will increase. Normally, a fault model is specified according to the underlying analysis or problem. The described model implements general fault and value functions which gives the ability to generalize. Furthermore, the analyst is able to tune the model after the specification.

This work is structured as follows. Section 2 discusses related work. Section 3 introduces the fault model. Section 4 summarizes and concludes the paper. For the convenience of the reader, a list of acronyms is provided in Table 1.

Table 1: List of Acronyms and Symbols

Acronym	Meaning
α	Size of area, physical layer
AVF	Architectural Vulnerability Factor
c_i	Masking factor, component layer
CLB	Configurable Logic Block
f	Fault function
FIT	Failure in Time
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
ft	Flip-to
IOB	IO-Blocks
NoC	Network on Chip
SA	Stuck-At
SEU	Single Event Upset

2 Related Work

Most fault models try to abstract from the physical effects of a fault. Transient faults in the form of Single Event Upsets [3][4][5][6] can be simulated through flipping single bits in memory elements. According to Karlsson et al. [7] they can be separated in flip-to-0 (ft0) and flip-to-1 (ft1), data-, control-flow- and other faults. The transition probabilities $P\{0\rightarrow1\}$ and $P\{1\rightarrow0\}$ are assumed to be equal, proved by Karlsson et al. [8] as realistic. We do not restrict ourselves to bridge, open, delay, stuck-at or tran-

sient faults, since these consider one fixed location and area of a fault. For an extensive description of these faults and effects, please see e.g. [12]. E.g. a fault in form of an electric charge can exist in different time intervals and in different areas. Another approach to represent any general delay defect is the work of Patel et al. [13]. Although this abstraction brings many advantages, e.g. a reduced testing time for the circuit, it does not consider that a fault in an electronic circuit is much more analogue in such a way that a certain amount of unwanted current is produced or current is not forwarded or forwarded too late, e.g. due to masking effects.

With shrinking minimum feature sizes, the need to compute the vulnerability factor of a circuit arose. Mukherjee et al. [1] first introduced the term architectural vulnerability factor (AVF). The AVF is the probability that a visible failure will occur and corresponds to the effective fault rate. Unfortunately, a complete implementation (at least on a behavioral level) is needed to conduct an analysis. Recently, Li et al. [2] proposed a way to estimate the AVF online by using marginal modifications to the hardware. The method does not require any offline simulation or calibration for different workloads. The method presented here differs from both methodologies, since it is a more abstract and generic way to model faults. It is neither limited to a specific implementation, nor dependent on the execution of benchmarks, since our aim was to compute an average bound concerning the masking capability. The model does not differentiate strictly between permanent and transient faults, since the only difference is the duration. For the convenience of the reader we stick to the usual naming.

3 Fault Model

3.1 Physical Layer

First, fault effects on the physical layer are modeled. This bottom-up approach enables us to omit validation, since the physical effects are already validated through empirical studies.

On the physical layer, a die A is divided in different sections $\boldsymbol{A} = \begin{pmatrix} \mathbf{a_{1,1}} & \cdots & \mathbf{a_{n,1}} \\ \vdots & \ddots & \vdots \\ \mathbf{a_{1,m}} & \cdots & \mathbf{a_{n,m}} \end{pmatrix}$. We are considering multiple layers, used in modern CMOS designs. Figure 1 shows these layers by indicating a third dimension, formally with a third parameter, z with $\begin{pmatrix} \mathbf{a_{1,1,z}} & \cdots & \mathbf{a_{n,1,z}} \\ \vdots & \ddots & \vdots \\ \mathbf{a_{1,m,z}} & \cdots & \mathbf{a_{n,m,z}} \end{pmatrix}$. As granularity, we take the size of the areas $a_{i,j,k}$. It must be equal on all levels. The smallest granularity is the size of a transistor. Figure 1 shows the physical (implicit) and the component layer, consisting of components and the routing. To model layer interconnection faults, additional layers can be inserted. A component C_i can be as large as or be a multiple of a region $a_{i,j,k}$.

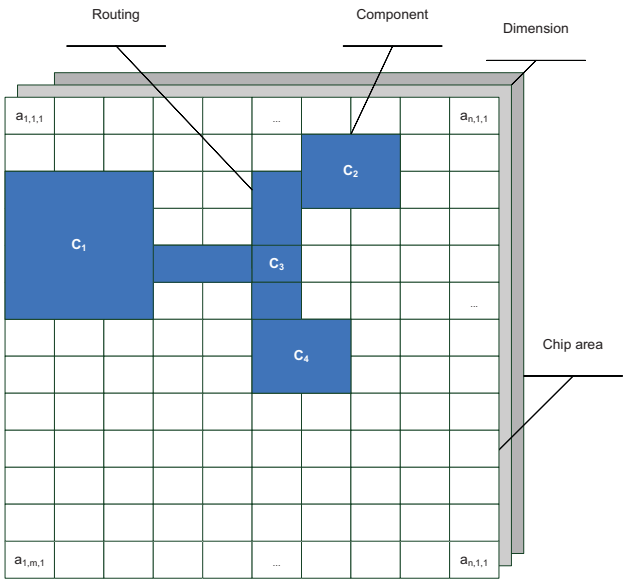


Figure 1: Fault model, physical and component layer

3.2 Fault and Value Functions

A fault function *f* models current over time. Transportation of current cannot occur in zero time. Thus, f is continuous and limited. If a fault is masked, a change at the inputs will not change the output behavior of the circuit. However, the inner state of the system can be faulty, e.g. maybe resulting in erroneous computations afterwards. For each area $a_{i,j,k}$, a function *f*, e.g. $\mathbf{f}: \mathbb{R}^5 \dashrightarrow \mathbb{R}, \mathbf{f_{a_{i,j,k}}}(\boldsymbol{\gamma}, \boldsymbol{\alpha}, \mathbf{I_0}, \mathbf{I_p(t)}, \boldsymbol{\tau})$ is applied. Multiple faults can be easily modeled by applying multiple functions to different areas. The properties of the exemplary fault function are listed in Table 2. The area α a fault can affect is dependent on the strength, duration, routing, latch masking effects etc.

Table 2: Fault Function Properties

Symbol	Meaning
τ (τ_a,τ_b)	Duration in ns
$I_p(t)$	Strength (current in mA over time t)
I_0	Starting current in mA
α	Area of $a_{i,j}$ in nm^2 ($\alpha_a\alpha_b$)
γ	Clocking probability (duty period)

Fault function properties like duration and strength can be used to model delay/ jitter faults, e.g. by limiting the duration of a (correct) signal. How can $I_p(t)$ be defined?

Example fault function – transient (SEU) fault

This depends on the technology and the fault scenario. In [6] the slow component τ_α of the current collecting dynamic of a SEU is active >0.5 ns in the substrate (maximum 0.5 mA), the quick component $\tau_\beta \leq 0.2$ ns with over 3 mA for a 0.6 µm-CMOS-process. We model this dynamic according to [9] with

$$\mathrm{I_p}(t) = I_0\left(e^{-\frac{t}{\tau_\alpha}} - e^{-\frac{t}{\tau_\beta}}\right).$$

The propagation of current is limited by many factors such as area, temperature and resistance.

Example fault function – permanent fault

The propagation of a fault is not always propagation of current, e.g. if we have a line stuck-at-0/1 fault. With $I_0=0$ or $\tau_\alpha=\infty$, we can also simulate such faults including timing behavior. Another simple function to model a permanent fault is $I_p(t) = k$ with $k \in \mathbb{R}$.

A value function models the uninfluenced way to propagate current. It is used to model logic '0' and '1' (and states in between). A fault function can be applied to every value function using appropriate operations, e.g. addition.

3.3 Component Layer

Each physical layer has one component layer. The elements on a component layer are components and routing, e.g. defined in the floorplan of the system. A system consists of $n \in \mathbb{N}$ multiple components C_i, $1 \leq i \leq n$ and connections between them. More formally, a net connects the individual components, where as a *net* is a triple $N = (C, R, \rightarrow)$, where *C* and *R* are disjoint sets of *components* and *routings*, and $\rightarrow \subset (C \times R) \cup (C \times R)$ is the *flow relation.* We first apply a masking model and define a masking factor $c_{i,j,k} \in \{0,1\}$ for each area element $a_{i,j,k}$, signaling if we have an element placed in the area $a_{i,j,k}$ ($c_{i,j,k}=1$) or not. Only these areas are considered during simulation. The values for every fault function distribute equally in each direction but can be e.g. changed by different sheet resistances. The fault will propagate only if ($c_{i,j,k}=1$). How much current will be produced/ propagated is determined on the physical layer by the fault and value functions. The maximum current is reduced by the sheet resistance in different regions.

3.4 Behavioral of components, case study

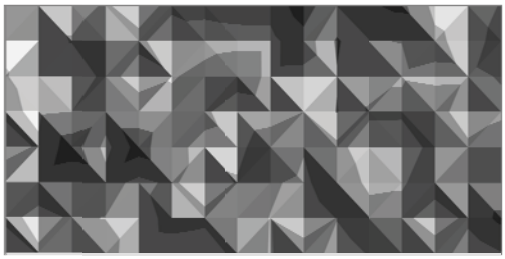
We consider discrete events for simulation [11]. The behavior of a component is modeled with a regular language that can be simulated with a FSM. We consider one behavioral example, a Field Programmable Gate Array (FPGA). We first regard within-die-variations (intra-process) on the physical layer which can be sub-divided into two main categories [10]: random and systematic. *Random* variations are small changes from transistor to transistor and are typically modeled with a normal distribution. *Systematic* variations exhibit high degrees of spatial correlation. Random variation causes changes in electrical parameters such as sheet resistance and threshold voltage. We consider the resistance since then the variation can be easily modeled. Figure 2a shows the result. In our case the smallest unit is α=1. The advantage of using α is that variations can be stretched over relevant units. The disadvantage is that if α is chosen inappropriately (too large or to small) the simulation results can be either be too detailed or inaccurate. Table 3 shows the simulation parameters and their association to the fault model level in detail.

Table 3: Simulation Parameters

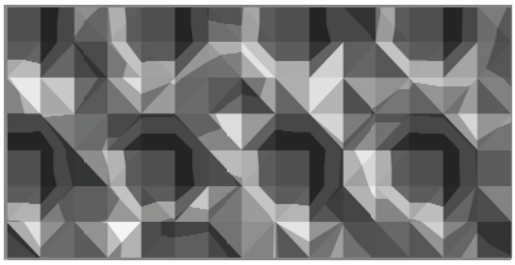
Model	Parameter	Value
Physical, L1	Die size	81×81 α
Physical, L1	α	1
Physical, L1	Process variation	Random
-	Device	FPGA, SRAM based
Component, L2	Simulation unit	CLB, switchboxes[1]
Component, L2	Layers	1
Physical, L1	Fault, time unit	ns
Physical, L1	Fault duration	[0…20] (ns)
Physical, L1	Simulated faults	Transient, permanent, see Figure 2
Physical, L1	Fault location	Random on die
Physical, L1	Simulation memory	Faults deleted after occurrence
-	Distribution	Exponential
-	#Input vectors	10^6
Physical, L1	Fault rate	10^{-5}
Physical, L1	Clock duty	50%
Physical, L1	Clock cycle	1 ns

To keep the analysis simple, a die size of 81×81 α and a random process variation were considered. The component mask was applied. Figure 2b shows the results for a section of 8×16 α. The darker areas are sections where no components are placed. Remember that we regard the raw FPGA – therefore the function of the CLBs is not configured and switchboxes are always connected with their neighboring CLBs. According to [14] fully segmented interconnections are uncommonly used in an FPGA with a high number of CLBs, because performance (in terms of propagation delay) is dependent on the number of traversed switchboxes. Thus, this assumption is realistic.

[1] "Switchbox" is used synonymously for "Programmable Interconnect Points".



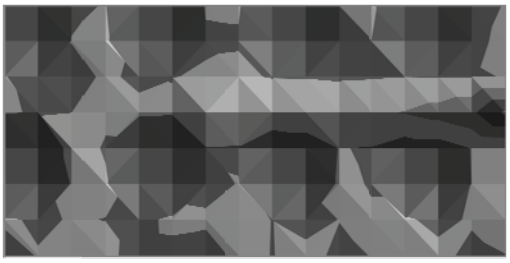
a) Random process variation, sheet resistance



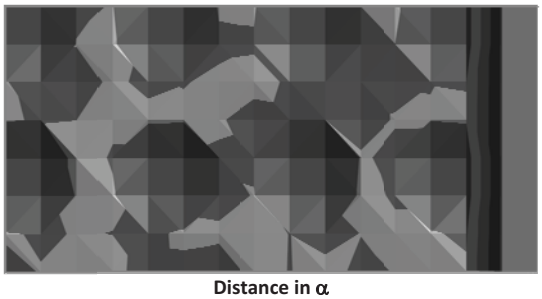
b) Component mask applied, sheet resistance

Figure 2: Random process variation and masking

Two possible fault scenarios, the transient and the permanent case are shown in Figure 3a and b by taking the component mask results from Figure 2b as starting point. The (random sheet) resistance in each relevant block of size α^2 will either lead to a better or worse propagation of the fault. Remember that the fault only travels along the non-masked components, thus simulation time will be reduced.



a) Component mask applied, transient pulse (right)



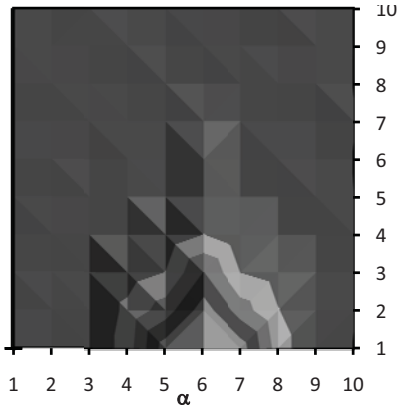
b) Component mask applied, SA1 fault (top-bottom)

Figure 3: Transient and permanent fault

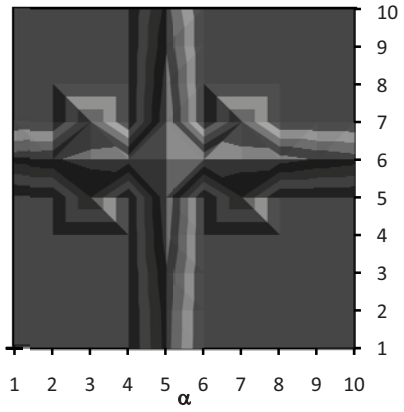
3.5 Behavioral level

For the behavioral simulation, we developed a multi-FSM simulator, consisting of CLB and switchboxes, specified in VHDL. Since only hints are available from manufacturers, switchboxes were implemented by using crossbars. We first simulated a small FPGA, consisting of 10 × 10 CLBs, 40 IOBs and 9 × 9 switchboxes, to measure the initial performance of the simulator. During simulation, transient faults were injected with rate $\varphi=10^{-5}$ per simulation run. As the smallest timing unit is ns, the result is a (purposely) accelerated fault rate. As inputs for the pins and respectively the IOBs, a random input was generated (10^6 vectors). To get independent results, we injected one fault at a time for the specified maximum duration of a fault (20 ns) into the design.

The MTTF of a section, a component or the whole circuit can be calculated by averaging the fault propagation over all areas in that section. The results for a 9 × 9 α section are shown in Figure 5a and b. For simplicity we assume that faults are always forwarded. Both figures show the case of a transient fault at different locations (CLB and switchbox) and same strength. A switchbox easily propagates a fault across the whole section. At first sight, this confirms the result from [21], since their implementations need about 10 times more routing resources and thus the error rate concerning the routing is about 10 times higher. These results consider a specific implementation, going up to the behavioral level. The conducted simulations consider the physical, routing and behavioral level. Additionally, by regarding the work of Héron et al. [22], we see from their reliability figures that the reliability of a design is directly related to the number of switchboxes (#ON-switches).



a) CLB bypass configuration



b) Switchbox, 4 neighbors

Figure 4: Transient fault, CLB and switchbox

Unfortunately, there are only hints concerning the error rate from manufacturers for specific parts, i.e. switchboxes. Some are given in [15] with details in [16]. From these and the device data some conclusions can be drawn. We did select devices, since some data of [15] is (astonishingly) not listed in [16]. Table 4 shows the real time soft error rates for the selected technologies. Configuration RAM (Conf.) comprises switch boxes and slices[2]. From [20] we know that a switchbox can realize 4 to 6 simultaneous connections for each pin (16-20). Thus, 64 to 120 connections can be made in total.

Table 4: Real time soft error rates (from [15])

nm	Neutron cross-section per bit		FIT/Mb		Device Data [17][18][19]	
Tech	**Conf**	**Block RAM**	**Conf**	**Block RAM**	**Slices**	**IO**
150	$2.56*10^{-14}$	$2.64*10^{-14}$	405	478	33792	1104
130	$2.74*10^{-14}$	$3.91*10^{-14}$	437	770	23616	852
65	$6.70*10^{-15}$	$3.96*10^{-14}$	163	676	17280	800

From [20] the amount of flip-flops used in CLBs (2+16 [4 inputs]) and IOBs (2) can be extracted. Since 64 to 120 connections have to be routed, an equal amount of flip-flops is needed (without considering any improvements, e.g. connections made with pass-transistors), confirming switchboxes as a main source for faults.

4 Conclusion and Future Work

In this paper, we presented a theoretical model to compute the fault propagation on different levels. It is separated into three parts, the physical, component and the behavioral level. On the physical level, transient or permanent faults can be modeled with different fault functions. By using a mathematical model of faults, an arbitrary number of faults and effects can be used. On the component level, the floorplan is used to mask out areas which are of no interest, e.g. without logic or routing. The model can be continuously used from the first planning until implementation. The simulation is done in O(n) time in relation to the total number of CLBs. Multiple faults can be inserted and their propagation be measured at any point in time or at any location. From the experimental results, we concluded and substantiated that switchboxes contribute and propagate much more faults than CLBs. A crucial point is simulation time. Components can be pre-simulated and used, if no fault is injected in them. Simulation time can be further reduced by an appropriate selection of α. A future implementation of the simulator will consider parallelization on the physical layer.

[2] „Slice“ and “CLB” are synonymously used.

References

[1] S.S. Mukherjee, C.T. Weaver, J. Emer, S.K. Reinhardt, T. Austin. *Measuring architectural vulnerability factors*. Micro, IEEE Volume 23, Issue 6, Nov.-Dec. 2003 pp. 70 – 75.

[2] X. Li, S. Adve, P. Bose, J. A. Rivers. *Online Estimation of Architectural Vulnerability Factor for Soft Errors*. In Proc. of 35th International Symposium on Computer Architecture (ISCA '08), June, 2008.

[3] D. Alexandrescu, L. Anghel, M. Nicolaidis. *New methods for evaluating the impact of single event transients in VDSM ICs*. Defect and Fault Tolerance in VLSI Systems, 2002. DFT 2002. Proceedings. 17th IEEE International Symposium, pp. 99 – 107, 2002.

[4] E. Normand. *Single-Event Upset at Ground Level*, IEEE Transactions on Nuclear Science, vol. 43, no. 6, 1996, pp. 2742-2750.

[5] E. L. Peterson. *Single-Event Analysis and Prediction*, IEEE Nuclear and Space Radiation Effects Conference, Short Course, 1997.

[6] T. Karnik et al. *Characterization of Soft Errors caused by Single-Event Upsets in CMOS Processes*, IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 2, pp. 128-143, 2004. DOI: 10.1109/TDSC.2004.14.

[7] J. Karlsson, P. Lidén. *Transient Fault Effects in the MC6809E 8 Bit Microprocessor: A Comparison of Results of Physical and Simulated Fault Injection Experiments*, Technical Report No. 96, Dep. Of Computer Engineering, Chalmers University of Technology, Göteborg, Sweden, 1990.

[8] P. Lidén, P. Dahlgren, R. Johansson, J. Karlsson. *On latching probability of particle induced transients in combinatorial networks*. In Proc. of the 24th Int'l. Symp. On Fault—Tolerant Computing, pp. 340-349, 1994.

[9] G.C. Messenger, *Collection of Charge on Junction Nodes from Ion Tracks*. IEEE Trans. on Nuclear Science, Band NS-29, S. 2024-2031, 1982.

[10] E. Humenay, D. Tarjan, and K. Skadron. *Impact of Process Variations on Multicore Performance Symmetry*. In Proc. of Design, Automation and Test in Europe Conference 2007 (DATE), April 2007.

[11] C. Cassandras, S. Lafortune. *Introduction to discrete event systems.* Kluwer Academic Publishers, 1999.

[12] M. J. Howes and D. V. Morgan. *Reliability and Degradation - Semiconductor Devices and Circuits,* Wiley, 1981.
[13] K. Heragu, J.H. Patel, V.D. Agrawal. *Segment Delay Faults*: A New Fault Model. In Proc. of 14th VLSI Test Symposium, 1996.
[14] S. Pontarelli, M. Ottavi, V. Vankamamidi, A. Salsano, F. Lombardi. *Reliability Evaluation of Repairable/Reconfigurable FPGAs.* Defect and Fault-Tolerance in VLSI Systems, IEEE International Symposium on, pp. 227-235, 21st IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT'06), 2006.
[15] Xilinx Inc. *Device Reliability Report*. Second Quarter 2010, http://www.xilinx.com/support/documentation/user_guides/ug116.pdf, 2010.
[16] A. Lesea, Xilinx Inc. *Continuing Experiments of Atmospheric Neutron Effects on Deep Submicron Integrated Circuits.* http://www.xilinx.com/support/documentation/white_papers/wp286.pdf
[17] Xilinx Inc. *Virtex™-II Platform FPGAs: Complete Data Sheet*, DS031, October 2003.
[18] Xilinx Inc. *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet*, DS083, November 2007.
[19] Xilinx Inc. *Virtex-5 Family Overview*, DS100, February 2009.
[20] Xilinx Inc. *XC3000 Series Field Programmable Gate Arrays (XC3000A/L, XC3100A/L)*, 1998.
[21] M. Sonza Reorda, L. Sterpone, M. Violante. *Efficient Estimation of SEU Effects in SRAM-Based FPGAs.* pp. 54-59, 11th IEEE International On-Line Testing Symposium, 2005.
[22] O. Héron, T. Arnaout, H.-J. Wunderlich. *On the Reliability Evaluation of SRAM-Based FPGA Designs*. In Proc. of the 15th IEEE International Conference on Field Programmable Logic and Applications (FPL'05), pp. 403-408; August 2005, Tampere, Finland.