Dynamic delay-fault injection for reconfigurable hardware

Bernhard Fechner
Lehrgebiet VLSI und Parallelität
FernUniversität Hagen
58084 Hagen
Bernhard.Fechner@fernuni-hagen.de

Abstract

Modern internet and telephone switches consist of numerous VLSI-circuits operating at high frequencies to handle high bandwidths. It is beyond question that such systems must contain mechanisms making them reliable through fault-detection or fault-tolerance. For monetary reasons, one or multiple Field Programmable Gate Arrays (FPGAs) are used in modern Application Specific Integrated Circuit (ASIC) development systems before production. Hardware manufacturers have a strong focus on quick fault-injection to verify and validate the correct function of such a (fault-tolerant) system. However, current FPGA-based fault injection schemes do not consider delay faults. In this paper we present an extension to traditional FPGA fault injection schemes without any additional hardware for fixed and small hardware overhead for dynamic phase shifting. By using digital clock managers (DCMs), we are able to inject delay faults very fast through phase-shift variation of the clock without reconfiguring the FPGA.

1. Introduction

As the minimum feature size of integrated circuits shrinks, a phenomenon which was only known from aerospace applications has to be considered: The increasing probability that heavy-ion or high-energetic α-particles will hit one or more transistors or capacitors, causing mainly transient errors and seldom permanent failures. Silicon pollution will lead to more timing-related faulty behavior (e.g. because of electromigration, CMOS switching times). The decreasing electrical potential between '0' and '1' and the increasing clock frequency lead to more signal-related delay faults. Furthermore, fluctuations on power nets will cause a temporary performance loss of the concerned circuits. For these reasons, designers of future CMOS-based computer systems must take delay faults even more into account. Thus, there is a great need for delay-fault injection to test these designs. This work shows how to extend the limitation of FPGA-based fault injection on stuck-at and SEU-faults to delay faults with minimal hardware overhead.
The paper is organized as follows: In Section 2 related work is described; Section 3 presents the fault-model. Section 4 describes the fault-injection in detail. Section 5 concludes the paper.

2. Related work

Dynamic fault injection using dedicated additional hardware which allowed the injection of different faults without reconfiguring the FPGA, was proposed in [5][6][7]. A major limitation of this technique is that hardware has to be added to the design. The additional hardware increases with the number of faults to inject, which limits the size of the circuits that can be simulated. In [8][10][13] faults are injected only in Look-Up-Tables (LUTs), which reduces the fault-type spectrum and thus the quality of fault injection, because most FPGA-designs do not only use LUTs. Recently, other hardware fault injection approaches were proposed in [9] using the JBITS [10] interface for partial FPGA reconfiguration. Additional to faulty LUTs, JBITS can provoke erroneous register values [9] but requires the Java SDK 1.2.2 [11] and the XHWIF [12] hardware interface.

3. The Fault Model

The *Stuck-at Fault Model* is the most common and general fault model for permanent logical faults. It assumes that a circuit fault manifests itself through the effect that one or more circuit nodes are stuck at 0 or 1 (SA01). In Table 1 we distinguish Line Stuck-At (LSA01) and Transistor Stuck-At (TSA01) faults. Sin-

gle Event Upsets (SEUs) are transient errors which are caused by high-energetic α-particles hitting the die. SEUs are modeled by bit-flips of the corresponding latches or memory cells. Although simple, it matches closely the real faulty behavior [5]. Single Event Disturbances (SEDs) cause temporal disturbance of digital information. Effects e.g. caused by electro-migration or higher resistance in CMOS wiring are hard to model. In some cases, the circuit works correctly but slower. In others, the faulty behavior manifests only at certain frequencies or input vectors. Some defects lead to a delay of a 0-1, 1-0 transition and result in the effect that the circuit is not able to keep its timing specification. The model for these dynamic faults is called *delay fault model*. There are two main delay fault models: The *gate-delay* model [1][2][3] where a delay fault manifests through delayed gate transitions and the *path-delay* model [4][5] where a fault occurs if the propagation delay along a path in the circuit is greater than the specified limit. In this paper we focused on a mixture of the *transient*[1] *bit-flip* model, which results in the modification of the content of a storage cell, the *stuck-at* fault model, supporting Line Stuck-At (LSA) and Transistor Stuck-At (TSA) faults and the *path-delay* fault model. A (path-) delay fault occurs iff $|t(V_2) - t(V_1)| > s$, where $t(V_1)$ is the time a circuit has stabilized under test vector V_1 which was applied at time *t-i*. $t(V_2)$ represents the time the test vector V_2 is applied after the circuit has stabilized under V_1. A delay fault is detected if the final value of the transition, propagating from circuit input to circuit output, can not be measured within the operational clock interval *s*, meaning that propagation (transition) delays along a path fall outside the specified timing limit.

4. Description of Fault Injection

Table 1 shows a list of reconfigurable FPGA components. A hook means that the specified fault can be injected in the corresponding component. Configurable Logic Blocks (CLBs) are the building blocks for implementing custom logic in every FPGA. Each CLB has two slices. Each slice has two 4 input LUTs, 2 flip-flops (which can also be configured as latches) and some inaccessible carry logic. We did not mention the DCM for producing delay faults, because it is the only component which is able to induce this kind of errors[2]. The cases that a LUT is configured as logic, (S)RAM or ROM are handled separately as LUT/ Distributed RAM/ROM. LSA faults in switch boxes are assumed to be collapsed to faults at lines of logic elements.

Table 1: Bitfile-accessible FPGA-components

	Injected Fault Type			
Component	**Transient**		**Permanent**	
	SEU	SED	TSA01	LSA01
CLB	✓		✓	
LUT			✓	
Distr. RAM	✓		✓	
ROM			✓	
BlockRAM	✓		✓	
Switch Box		✓		✓

Before fault-injection can be done, it is necessary that a healthy (assumed fault-free) FPGA configuration (bitfile) is uploaded. Therefore, the hardware description is synthesized and mapped to a specific FPGA device. The principle of FPGA fault-injection is shown in Figure 1:

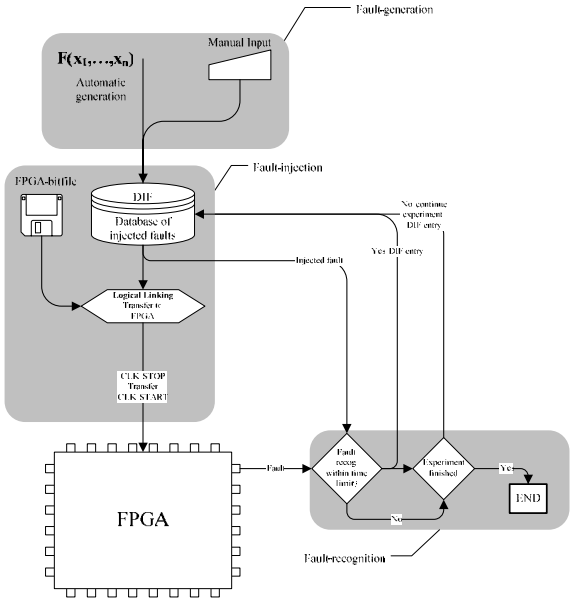

Figure 1: Principle of FPGA fault-injection

- First, we specify the number of faults (1-n), their duration (1-∞), thus the fault-type (transient or permanent) and the duration of the experiment (1-n). The specification can be done automatically or manually. The list of faults is stored in the *Database of Injected Faults (DIF).* For automatic fault-generation either pre-collected mission data or a functional description of the error/time behavior can be used.
- Second, we do a logical linking between the generated error list and the bitfile, containing a

[1] We do not distinguish transient and intermittend faults.

[2] Some delay-faults can be provoked by injecting SEUs in global clock lines.

healthy FPGA configuration. Then we stop the FPGA-board clock and upload the changed (faulty) bitfile to the FPGA. After the transfer, we start the clock again. Although it is possible to use the FPGA-programming modes for clock control, this could cause problems, because other on-board components could continue their work (e.g. because they use another clock), relying on results from the FPGA. We generally suggest stopping the board clock. For fast injection of transient or permanent faults, only the parts containing faulty information can be uploaded (partial reconfiguration).

– Third, we specify the maximum time (1-n clock cycles) between fault at time t, F(t) and result R(u) (a faulty behavior) at time u. This is done by the module *fault-recognition*. The module knows the contents of the DIF. If an injected fault is detected or corrected by the fault-tolerance mechanisms in hardware (supposing that the FPGA signals the error externally e.g. with different pin-value combinations) within the specified time, the information |u-t| is stored in the DIF, else '∞' is stored. The output data of each fault injection experiment is analyzed and faults are categorized according to their effects. Fault collapsing is performed when different faults result in identical faulty configurations.

Modern Xilinx FPGA families like Spartan-3 [16] and Virtex II pro [17] have multiple on-chip digital clock managers (DCMs). DCMs provide clock management features such as clock de-skew, frequency synthesis and phase shifting. The basic idea of this work is to misuse the phase shift to produce non-timing conform circuit behavior. Furthermore, we combine this idea with conventional FPGA fault injection based on partial or total reconfiguration which was described above. A DCM can have two kinds of phase shift: fixed and variable mode. The minimal amount a signal can be phase shifted is dependent on voltage, current and technology. For Spartan-3 DCMs this is ~30-50ps [16]. It is possible to have different DCM-configurations by modifying the bitfile, since the DCM is part of the bitfile. The achieved speed may not always be sufficient. In fact, the DCM can be reconfigured very fast on-line without doing a total or partial reconfiguration. This is called dynamic phase shift adjustment [14]. Assuming a 100MHz clock, we have 600µs for a minimal partial reconfiguration [15] with 16 CLBs and 1.03µs of a DCM reconfiguration (103 clock cycles for a Spartan-3[16]). Figure 2 shows the timing diagram for a dynamic phase shift reconfiguration. When the signal PSDONE goes active for one cycle, the DCM has completed the adjustment. It may require maximal 100 CLKIN cycles plus three PSCLK cycles to effect a change [16]. Injection of timing-related errors will thus take maximal 103 clock cycles on Spartan-3 FPGAs.

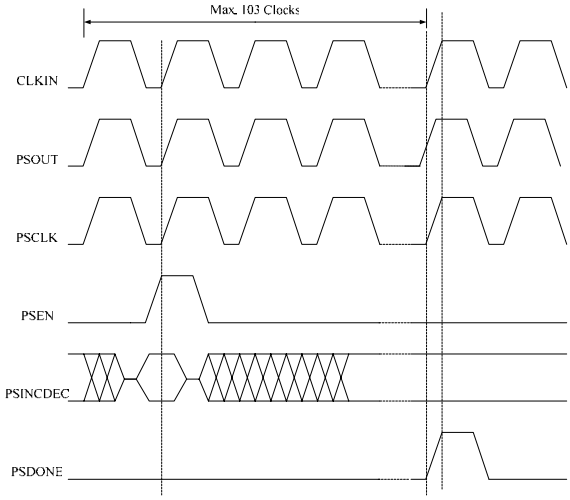

Figure 2: Dynamic phase shift reconfiguration

For dynamic phase reconfiguration, extra hardware for the error-dependent variation of the clock shift is necessary. This can be either an external part on the development board or internal FPGA hardware. To inject a delay fault, the outputs of the component which is in charge of producing such faults must be connected to the appropriate ports of the DCM (PSINCDEC, etc.). According to the delay-fault model, all supported delay faults have in common, that the response of the circuit under test does not appear within the specified timing frame at the circuit outputs. Thus, injected delay-faults do not cause a functional error but an incorrect timing behavior. The first effect we can produce with DCMs is a (positive or negative) clock *jitter* of all clocked components. A jitter occurs due to system noise and signal crosstalk, causing phase uncertainty. The result is ambiguity in the rising and falling edge of a signal. This can be done by a simple phase shift of the DCM, requiring no additional hardware if we use a fixed phase shift. To produce a variable phase shift, we either need an external component or FPGA resources to produce deterministic or random phase shifts. The second effect we can produce is an input dependent delay fault. This fault only occurs on a set of certain input vectors. To do this, we need one comparator, which is able to compare input vectors with pre-defined values which are stored in a memory. Figure 3 (left) shows the configurations described above. The third effect we can produce is a modification of the output data timing by a second DCM (Figure 3, right). A comparator can

be used to extend this method to be bit-vector dependent.

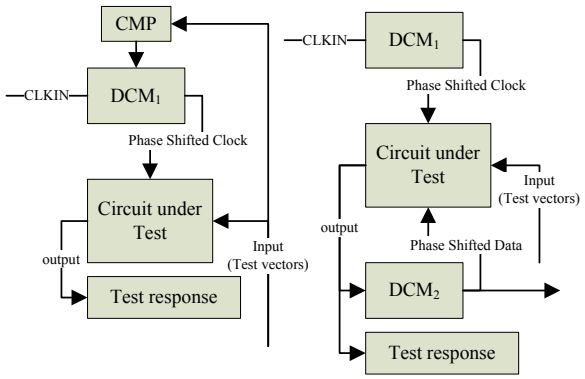

Figure 3: Possible configurations for fault-injection

5. Conclusion

In this paper we presented a new way to inject delay-faults in an FPGA-based design by using standard FPGA-components. Thus, the limitation of typical FPGA-based fault injection on certain fault-types (Stuck-at, SEUs) is broken. We are dependent on specific FPGA types (as every FPGA-based fault injection is) and the hardware can not be destroyed by the injector. Since DCMs exist only for dedicated FPGA families [16][17] we are even more restricted on hardware. With the help of the *Database of Injected Faults* we are able to inject multiple transient and permanent faults. They can be injected into a non-configured area of the FPGA and have no effect. If we compare the time for a DCM configuration with the time for a minimal partial reconfiguration (16 CLBs), we see that the reconfiguration takes approximately 0.6ms, growing exponentially with the number of CLBs [15]. If we assume a 100MHz clock, we have 600µs for a minimal partial reconfiguration and 1.03µs of a DCM reconfiguration [16]. In [9] the injection times through reconfiguration for a fault with a development board were much higher (3.5s) than expected (100ms). For a DCM phase shift reconfiguration these times are of no concern, because we do not have to generate and upload a new bitfile.

References

[1] Y. Levendel, P.R. Menon, *Transition Faults in Combinatorial Logic Circuits. Input Transition Test Generation and Fault Simulation.* In Proc. of the 16th Fault Tolerant Computing Symposium, pp. 278-283, June 1985.

[2] J. Rajski, H. Cox, *A Method of Test Generation and Fault Diagnosis in Very Large Circuits.* In Proc. of the International Test Conference, pp. 932-943, Sept. 1987.

[3] J.A. Waicukauski, E. Lindbloom, B. Rosen, V. Iyengar, *Transient Fault Simulation by Parallel Pattern Single Fault Propagation.* In Proc. of the International Test Conference, pp. 542-549, Sept. 1986.

[4] G.L. Smith, *A Model for Delay Faults Based on Paths.* In Proc. of the International Test Conference, pp. 342-349, Sept. 1985.

[5] R.W. Wieler, Z. Zhang, R.D. McLeod, *Simulating static and dynamic faults in BIST structures with a FPGA based emulator*. In Proc. of IEEE International Workshop of Field-Programmable Logic and Application, pp. 240-250, 1994.

[6] S.A. Hwang, J.H. Hong, C.W. Wu, *Sequential Circuit Fault Simulation Using Logic Emulation*. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 17(8), pp. 724-736, August 1998.

[7] P.Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, M. Violante, *An FPGA-based approach for speeding-up Fault Injection campaigns on safety-critical circuits*. IEEE Journal of Electronic Testing Theory and Applications, 18(3), pp. 261-271, June 2002.

[8] M. Abramovici, P. Menon, *Fault Simulation on Reconfigurable Hardware*. IEEE Symposium on FPGAs for Custom Computing Machines, pp. 182-190, 1997.

[9] L. Antoni, R. Leveugle, B. Fehér, *Using Run-Time Reconfiguration for Fault Injection in Hardware Prototypes*. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 245-253, 2002.

[10] S. Guccione, D. Levi, P. Sundararajan, *JBITS: A Java-based Interface for Reconfigurable Computing*. In Proc. of the 2nd Military and Aerospace Applications of Programmable Devices and Technologies Conf., Sept. 1999.

[11] E. Lechner, S. Guccione, *The Java Environment for Reconfigurable Computing*. In Proc. of the 7th International Workshop on Field-Programmable Logic and Applications, pp. 284-293, Sept. 1997.

[12] P. Sundararajan, S. Guccione, D. Levi, *XHWIF: A portable hardware interface for reconfigurable computing*. In Proc. of Reconfigurable Technology: FPGAs and Reconfigurable Processors for Computing and Communications, SPIE 4525-20, pp. 155-160, June 2001.

[13] A. Parreira, J.P. Teixeira, M.B. Santos, *Built-in self-test preparation in FPGAs,* In Proc. of the 7th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, pp. 83-90, April 2004.

[14] Xilinx Inc., *Active Phase Alignment*, Application Note: Virtex II Series, Xilinx XAPP268, Dec. 2002.

[15] U. Malik, K. So, O. Diessel, *Resource-Aware Run-time Elaboration of Behavioral FPGA Specifications*, IEEE International Conference on Field-Programmable Technology, pp. 68-75, Dec. 2002.

[16] Xilinx Inc., *Spartan-3 FPGA Family: Complete Data Sheet*, Xilinx DS099, March 2004.

[17] Xilinx Inc., *Xilinx Virtex-II Pro and Virtex II Pro X Platform FPGAs: Complete Data Sheet*, Xilinx DS083, Nov. 2004.