Microprocessors and Microsystems 39 (2015) 1119–1129

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Closing the gap between speed and configurability of multi-bit fault emulation environments for security and safety–critical designs

Ralph Nyberg [a,*], Johann Heyszl [a], Dirk Rabe [b], Georg Sigl [c]

[a] Fraunhofer Institute AISEC, Parkring 4, 85748 Garching, Germany
[b] Hochschule Emden/Leer, FB Technik, Constantiaplatz 4, 26723 Emden, Germany
[c] Technische Universität München, EI SEC, Arcisstraße 21, 80333 Munich, Germany

ARTICLE INFO

Article history:
Received 13 January 2015
Revised 20 April 2015
Accepted 18 May 2015
Available online 29 May 2015

Keywords:
FPGA-based fault emulation
Multi-bit faults
Performance optimization
Configurability
Result evaluation

ABSTRACT

Steadily decreasing transistor sizes and new multi beam laser attacks lead to an increasing amount of multi-bit fault occurrences, e.g., during fault attacks against cryptographic implementations. Therefore, multi-bit fault injection becomes more important during security and safety verification. Fault injection techniques which are applicable during the development cycle of a device are based on either software implementations, e.g. formal methods and simulations, or fault emulation environments in hardware. So far, simulations provide the best configurability whereas fault emulation environments provide the best performance in terms of run time. This contribution presents an FPGA-based emulation environment that combines the advantages of both simulation-based and emulation-based environments. To the best of our knowledge, we are the first to achieve this. Permanent and transient multi-bit faults are configurable at run time where the selection of a fault model, the configuration of the injection time and fault duration is supported without the need for re-synthesizing the design. We propose three measures for performance optimization allowing us to support all the fault configuration capabilities at run time without performance penalty. Experimental results are provided for a hardened 8051-like microprocessor showing that the presented emulation environment reaches the theoretical optimal performance for a wide range of fault configurations using our proposed optimizations.

1. Introduction

Safety–critical circuits, e.g. in the application domain of aerospace, are exposed to alpha-radiation resulting in faulty behavior. Security devices, like smart cards, can be broken by deliberately injected faults which may result in leakage of secrets. Such an attack was shown by Boneh et al. [1] who exploited fault attacks to successfully break an RSA cryptoalgorithm. Fault countermeasures were introduced to security and safety–critical designs preventing the leakage of secrets and guaranteeing proper circuit behavior even in harsh environments. For example, fault countermeasures may detect illegal states and state transitions of finite state machines, illegal instructions or detect and correct faulty data on buses by utilizing error correcting codes. These fault countermeasures have to be evaluated during security and safety verification in order to check whether they work properly. Security and safety verification have to consider fault injection because fault countermeasures are only supposed to take any action in the presence of faults. There are different fault injection approaches which can be performed either post-silicon or pre-silicon. Design flaws detected post-silicon result in high design costs and prolong the overall development cycle. Therefore, fault injection approaches are needed which are applicable on HDL designs or netlists during the development cycle of a circuit. FPGA-based fault emulation fulfills this requirement. Furthermore, FPGA-based fault emulation is four orders of magnitude faster than simulation-based techniques [2] and the only real-time capable technique. Contrary to techniques based on simulations or formal methods, the size of the circuit under verification has no impact on the performance of FPGA-based fault emulation, except for the impact on the working frequency.

Physical attacks may affect more than a single gate. Therefore, fault injection approaches also have to consider multi-bit faults to model faults as close as possible to reality. Because of the large fault space, only a limited number of multi-bit faults can be modeled in feasible time. Relevant faults need to be chosen and configured individually. But with increasing numbers of faulty bits more data is needed for fault configurations. This data has to be uploaded over a communication interface which is the performance limiting factor

* Corresponding author.
E-mail address: ralph.nyberg@aisec.fraunhofer.de (R. Nyberg).

http://dx.doi.org/10.1016/j.micpro.2015.05.015
0141-9331/

of fault emulations [3]. In order to remove this limitation and as a result increase the performance of fault emulation further, autonomous fault generation on the FPGA was proposed. Autonomous fault generation avoids data interchange on the communication interface in between consecutive fault emulations. The performance of autonomous approaches is very close to the actual time needed by the Circuit Under Verification (CUV) to process the input which is the theoretical optimal performance. The major drawback of autonomous approaches is that these only support single-bit faults or randomized fault generation for multi-bit faults, completely removing the ability to configure individually chosen faults. This renders autonomous approaches useless for more complex multi-bit fault selection strategies, such as finding and configuring interesting multi-bit faults based on the layout of the CUV [4]. Therefore, in state-of-the-art fault emulation environments either the ability to configure individually chosen faults is removed or the performance gets worse for multi-bit faults. A decreasing performance limits the number of faults which can be modeled during verification even further.

We present an optimized, real-time capable, FPGA-based multi-bit fault emulation environment which is designed to maximize the emulation performance while configuration capabilities remain unaffected. Thus, the fault coverage is only limited by the size of the multi-bit fault space but neither by the configuration capabilities nor by the performance of our fault emulation environment. The fault model, the fault injection time as well as the fault duration can be configured for each single-bit and multi-bit fault without the need for re-synthesizing the design. The performance of the fault emulation environment is optimized by reducing idle times of the hardware in between consecutively executed fault emulations utilizing three different optimization measures. The presented optimization measures allow us to provide a user-friendly configurability at runtime without performance penalty for a wide range of multi-bit faults. In fact, the optimization measures shift the bottleneck from the communication interface to the actual time needed by the Circuit Under Verification (CUV) to process the input. So far, such a performance was only reached by autonomous fault generation on the FPGA. In contrast to our previous work [5], we also propose a new methodology allowing to tolerate differences in the timing behavior of fault experiments and reducing false positives for microprocessor-like designs, e.g. smart cards.

This paper is structured as follows. In Section 2 the related work is discussed. The proposed approach of our fault emulation environment is described briefly in Sections 3 and 4 outlines the details. A new methodology for evaluating fault emulation results of microprocessor-like designs preventing false positives is proposed in Section 5. Three performance optimization measures are outlined in Section 6. Performance results are presented in Section 7. Finally, Section 8 concludes this paper.

2. Related work

There are three different approaches to perform FPGA-based fault emulation: partial FPGA-reconfiguration, mutant- and saboteur-based modifications and instrumented circuit techniques at gate level. The following subsections discuss these approaches and also discuss necessities for applying the FPGA-based fault emulation to microprocessor-like designs.

2.1. Partial FPGA-reconfiguration

Exploiting FPGA-reconfiguration for fault grading was the initial fault emulation approach. Cheng et al. [6,7] proposed compile-time FPGA reconfiguration to model permanent faults based on the stuck-at model. Antoni et al. [8] proposed local (also known as partial or dynamic) real-time FPGA reconfiguration in order to also enable modeling transient faults. Basically, this methodology implements a read-modify-write scheme for the frame of the FPGA configuration RAM in which the fault is modeled. As shown by the results of a recent publication [9], the time overhead for modeling a single-bit fault is around 500 cycles (10 μs at 50 MHz clock). In case of modeling transient faults, the HW execution has to be interrupted in order to alter the HW behavior. Therefore, this methodology is considered to be slow and does not provide real-time capability for transient faults. We do not consider this approach since our goal is a fast, real-time capable environment for permanent and transient faults.

2.2. Mutant- and saboteur-based RTL modifications

Mutants and saboteurs enable fault injection capabilities by HDL modification. Mutants replace the original component RTL description by a description which is capable of fault injection. Saboteurs are components which are capable of altering signal behavior. Saboteurs are fault injection components which extend an RTL design without replacing the original description [10]. Mutants and saboteurs were originated in the RT level fault simulation domains [11,10]. Misera et al. [12] introduced mutants and saboteur to SystemC-based simulation and provide a good starting point for research on simulation-based fault injection techniques. Leveugle [13] utilized mutant generation for an FPGA-based emulation environment. Baraza et al. [14] present an emulation environment capable of placing saboteurs and generating mutants. Baraza et al. also propose an automatic saboteur placement and mutant generation in order to gain a better performance and Grinschgl et al. [15,16] implemented automatic saboteur placement. The advantage of mutant- and saboteur-based emulation environments is enabling early dependability analysis by using well known HDL resulting in high flexibility with respect to implementing different fault models. On the other hand, major drawbacks are the required HW-overhead and synthesis time overhead [15]. The high HW-overhead prevents enabling fault injection capability for every possible location at once. Therefore, multiple synthesis runs would be needed in order to provide a full coverage slowing down the over all performance drastically. Since we are aiming for a fast and complete solution for fault grading in netlists we do not consider this approach.

2.3. Instrumented circuit technique

Contrary to partial FPGA-reconfiguration and mutant- and saboteur-based RTL modifications, the instrumented circuit technique only requires a single time-consuming synthesis run and FPGA-configuration in order to provide complete fault coverage for multi-bit faults. Furthermore, the instrumented circuit technique does not require RTL or library modifications, is applicable for gate level netlists and is fast and real-time capable. Therefore, we consider this technique. Circuit instrumentation focuses on adding fault injection capability for flip flops (FFs) or combinational gates within a gate level description. In order to keep the amount of fault locations for a given Circuit Under Verification (CUV) low, we consider circuit instrumentation techniques for FFs only. Dependent on a certain fault model, FFs are extended by additional logic in order to support fault injection. Either a single FF (single-bit fault) or a set of FFs (multi-bit fault) is selected to be faulty in a particular fault experiment and all the other FFs keep their fault free functionality. Civera et al. [17,18] propose a register for FF selection, the so called fault mask register, implemented as scan-chain. When the fault injection time is reached, all faults selected in the fault mask register are enabled for one clock cycle. For each fault experiment the entire content of the fault mask register is uploaded over