Soft Error Sensitivity Evaluation of Microprocessors by Multilevel Emulation-Based Fault Injection

L. Entrena, M. García-Valderas, R. Fernández-Cardenal, A. Lindoso, M. Portela García, and C. López-Ongil

Abstract—Estimation of soft error sensitivity is crucial in order to devise optimal mitigation solutions that can satisfy reliability requirements with reduced impact on area, performance, and power consumption. In particular, the estimation of Single Event Transient (SET) effects for complex systems that include a microprocessor is challenging, due to the huge potential number of different faults and effects that must be considered, and the delay-dependent nature of SET effects. In this paper, we propose a multilevel FPGA emulation-based fault injection approach for evaluation of SET effects called AMUSE (Autonomous MUltilevel emulation system for Soft Error evaluation). This approach integrates Gate level and Register-Transfer level models of the circuit under test in a FPGA and is able to switch to the appropriate model as needed during emulation. Fault injection is performed at the Gate level, which provides delay accuracy, while fault propagation across clock cycles is performed at the Register-Transfer level for higher performance. Experimental results demonstrate that AMUSE can emulate soft error effects for complex circuits including microprocessors and memories, considering the real delays of an ASIC technology, and support massive fault injection campaigns, in the order of tens of millions of faults within acceptable time.

Index Terms—Soft error, single event transient, single event upset, fault injection, FPGA emulation.

1 Introduction

Soft errors have become a concern for critical applications where dependability must be ensured. As manufacturing technology progresses by reducing feature size, providing more integration density and increasing device functionality, devices are more sensitive to soft errors [1]. In this context, techniques and tools to efficiently evaluate the soft error sensitivity of complex circuits, such as microprocessors, are strongly needed.

Single Event Upsets (SEUs) that change the state of a memory cell or latch have traditionally been considered the most important type of soft error. However, soft errors can be produced as well in combinational logic, which are known as Single Event Transients (SETs). The analysis of SET effects is challenging, as SET effects depend on circuit delays and therefore the analysis must be done at a lower abstraction level. On the other hand, the potential variety of SET effects is huge, as SETs may occur at any gate at any time and propagate to multiple memory elements.

Many techniques for detecting, locating, recovering, and tolerating temporary or permanent errors are available [2], as this has been an area of active research over decades. However, the direct application of these techniques generally produces an unacceptable impact in area, performance, and power consumption. For instance, the well-known Triple Modular Redundancy (TMR) technique is able to mask any single fault but introduces more than 200 percent area overhead. Software redundancy approaches [3] also introduce similar performance degradation figures.

Optimal mitigation solutions must be based on selectively applying fault tolerance techniques just to the most critical parts of the design in order to minimize the impact in area, performance, and power consumption. To this purpose, effective tools for soft error sensitivity analysis are needed. These tools must be able to work early in the design cycle in order to identify the critical parts of a design that must be hardened.

Fault injection is a widely accepted method to evaluate soft error effects. Simulation-based fault injection [4] uses a simulation tool to inject and propagate faults in a design model. However, simulation-based fault injection is quite slow. It must be noted that identifying the critical components of a design is a much more complex objective than simply estimating the Soft Error Rate (SER) for the whole circuit. Every component in the circuit must be individually assessed by injecting a significant number of faults. Therefore, very large fault injection campaigns must be performed. In order to accelerate the fault injection process, emulation-based fault injection approaches have been developed in recent years [5], [6], [7], [8], [9]. These methods use FPGAs to prototype the Circuit Under Test (CUT) and support the fault injection mechanisms.

Microprocessors are at the heart of any digital system today. A soft error in a microprocessor may produce a wrong computation result or losing the control of a system with catastrophic consequences. On the other hand, microprocessors can have a huge variety of failure modes

• *The authors are with the Department of Electronic Technology, University Carlos III of Madrid, Av. de la Universidad 30, Leganés 28911, Madrid, Spain. E-mail: {entrena, rfcarden, celia}@ing.uc3m.es, {mgvalder, alindoso, mportela}@ing.uc3m.es.*

Manuscript received 25 Jan. 2010; revised 22 July 2010; accepted 17 Nov. 2010; published online 7 Dec. 2010.
Recommended for acceptance by D. Gizopoulos.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-2010-01-0053.
Digital Object Identifier no. 10.1109/TC.2010.262.

depending on the application. For this reason, the evaluation of these failure modes for the particular application considered is crucial in order to devise optimal mitigation strategies. Emulation-based fault injection is a well-suited approach for this task. However, this requires specific support for emulation of memories, which is not commonly available [8], [10].

This paper proposes an efficient emulation-based fault injection approach for complex circuits including microprocessors. This approach is primarily intended for SETs, i.e., transient faults at the output of logic gates, but it also supports SEUs as a particular case of SETs. Several emulation-based fault injection approaches for SEUs have been proposed [5], [6], [7], [8], [9] and some of them have been used on microprocessors [11]. However, SET fault injection is much more difficult [12], [13]. To the best of our knowledge, this is the first solution proposed for evaluation of SET effects in complex circuits that is entirely based on FPGA emulation.

Evaluation of SET effects requires a Gate Level (GL) model that includes delay information. Recently, efficient and accurate SET emulation approaches have been proposed using a quantized model [14], [15]. However, these approaches have just been demonstrated for small combinational circuits. This can be inefficient for a microprocessor, as a SET may produce state changes that manifest after many clock cycles.

To accelerate the propagation of SET effects across clock cycles, we use a Register-Transfer level (RTL) model of the microprocessor. The Gate level and Register-Transfer level models of the microprocessor under test are integrated seamlessly in the FPGA to provide fast and accurate fault analysis.

The approach proposed in this paper integrates and adapts existing techniques in a new original framework. The result is a unique and general emulation-based fault injection system, called Autonomous MUltilevel emulation system for Soft Error evaluation (AMUSE), with the following main contributions:

1. SET and SEU fault injection capabilities.
2. Consideration of real delays, as produced by the synthesis tool for any target ASIC technology for the circuit under test, and including electrical masking effects.
3. Support for complex circuits, such as microprocessors, that include memories and require large execution workloads.
4. Fast and accurate emulation by performing fault injection tasks at the appropriate abstraction level.

The remaining of the paper is as follows: Section 2 presents the background and analysis of the state-of-the-art in the topic. Section 3 describes the multilevel emulation-based fault injection approach. The Gate level and the Register-Transfer level models are described in Sections 4 and 5, respectively. Section 6 describes the emulation controller and the operation of the system. Section 7 shows how the system is built for an application. Finally, Section 8 presents the experimental results and Section 9 summarizes the conclusions of this work.

2 Background and Previous Work

Fault injection is a widely accepted method for the evaluation of circuit sensitivity regarding Single Event Effects (SEE), in particular Single Event Upsets and Single Event Transients.

The SEU effect is commonly modeled by the bit-flip model applied to circuit Flip-Flops (FFs). When a SEU affects a flip-flop, its value is flipped, that is, it is loaded with the opposite value. The evaluation consists in analyzing the effect of flipping a flip-flop value on the circuit behavior. This is usually accomplished by comparing the results produced by the faulty circuit and the fault free circuit (called golden circuit).

Evaluation can be carried out by simulation. A simulator is used to analyze the circuit behavior once a fault has been injected (a flip-flop value has been changed). The result can be different if the fault is injected in a different location (flip-flop) or a different time instant (clock cycle). Many faults must be injected to obtain representative sensitivity results, so the same simulation must be repeated many times.

The SEU evaluation problem is bidimensional (location and time). For example, for a circuit with 3,000 flip-flops and a common workload of 10,000 clock cycles, there are 30 million possible faults. Simulation has proven to be too slow to test even a fraction of so many faults in a reasonable time. To solve this problem, FPGA emulation is used.

In FPGA emulation, a programmable circuit (the FPGA) is used to replace the simulator. The circuit is implemented using the FPGA, which can emulate the circuit behavior at the RTL. So, instead of using a simulator, a much faster hardware circuit is used. The mechanisms to inject faults and controlling the process are more complex, but the speed improvement makes it possible to analyze a real circuit in a reasonable time.

There are mainly two techniques to perform fault injection in FPGA emulation. Using FPGA reconfiguration mechanisms [6], [7], a fault is injected by loading a new reconfiguration bitstream into the FPGA which corresponds to the faulty circuit. Even though partial reconfiguration can be used to reduce the size of the bitstream, this is a slow process. Alternatively, circuit instrumentation can be used for fault injection [8], [9]. Circuit instrumentation consists in inserting some pieces of hardware, also called instruments, which can provide external controllability and observability to inject a fault and observe its effects. Circuit instrumentation is an automatic process, that is basically performed by substituting cells of the CUT by their equivalent instrumented cells. Then, the instrumented circuit is prototyped in the FPGA.

In previous works, the authors have already proposed an enhanced variation of the instrumented circuit technique called Autonomous Emulation. In the original and subsequent versions of the instrument circuit implementations, there is always a continuous interaction between the emulation circuit and the host computer controlling the process. In the Autonomous Emulation paradigm, every required functionality is implemented in the FPGA along with the circuit model, so that no FPGA-host interaction is needed and the fault injection speed is improved by three orders of magnitude [9]. Fig. 1 shows the structure of an Autonomous Emulation system.

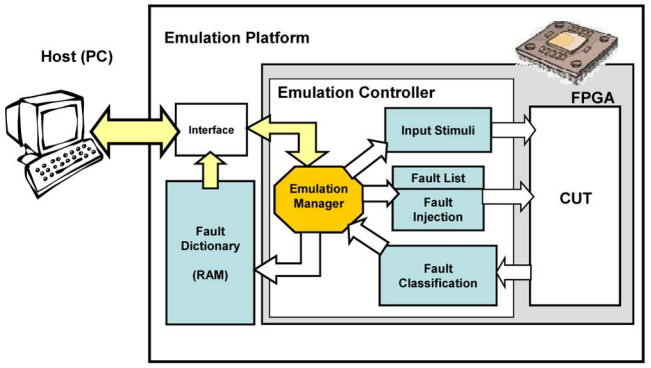

Fig. 1. Autonomous Emulation system.

While the SEU evaluation problem is already solved by emulation-based fault injection, the single event transient evaluation problem is not. The SET evaluation problem is far more complex [17].

A SET can be modeled by a transient bit-flip in the output of a gate. The duration of the bit-flip depends on the energy of the particle striking the circuit. Once the particle has hit a gate, the flipped pulse may propagate through the combinational logic and eventually get stored in one or several circuit flip-flops, producing a SEU or a Multiple Bit Upset (MBU).

Evaluating SET faults is more complex than SEU faults in two senses. On the one hand, there are many more possible SETs than SEUs, regarding location, time instant, and pulse length. A SET can strike different gates, at different time instants and the pulse produced can have different length depending on the particle energy.

On the other hand, to analyze the effect of SETs in the circuit, propagation delays must be taken into account in the simulation, in order to calculate the pulse propagation, and there are three masking effects that can prevent a transient pulse from propagating and being latched by a memory element: logic masking, latch window masking, and electrical masking [16]. Logic masking occurs when the SET is not located in a sensitized path and thus cannot be propagated. Latch window masking occurs when the transient reaches the sequential elements outside of their latching time window. Electrical masking occurs when the transient is attenuated by subsequent logic gates until it is eventually filtered. A logic simulator can take into account logic masking and latch window masking, but to consider electrical masking, an electrical simulator is required. In addition, new dynamic propagation effects such as Propagation Induced Pulse Broadening (PIPB) [20], [21] have recently been found and are being investigated.

The huge amount of possible faults and the need to use an electrical simulator makes simulation a very inefficient method to obtain SET sensitivity results. Probabilistic considerations [17] or fault collapsing at the RTL [4], [22] may be used to reduce the amount of faults to inject. The speed of hardware emulation techniques is very attractive for SET evaluation, but delay information must be embedded in the circuit model implemented in the FPGA. Just synthesizing the CUT for an FPGA, as it is done for SEU evaluation, would produce an equivalent functional circuit model but with different propagation delays.

Existing approaches for SET emulation are based on embedding timing information in some way, such as the topology of the circuit [12]. Later approaches use a combination of gate level simulation with RTL emulation [13]. In this case, gate level simulation is first used to evaluate the propagation of SETs across combinational logic, and then the analysis is completed by transferring the results to an emulation platform and injecting them using FPGA reconfiguration mechanisms. This approach does not solve the problem of a slow gate level simulation.

Recently, efficient and accurate SET emulation approaches have been proposed using a quantized model [14], [15]. Quantized CUT delays are implemented using shift registers [14] or nonlinear counters [15] that can be mapped in a FPGA. These approaches provide very fast fault injection rates with accuracy close to analog simulation. The solution proposed in this work follows these approaches and the speed offered by Autonomous Emulation to build an emulation solution that first analyzes the propagation of a SET throughout the logic up to the flip-flops, and then analyses the effect of the bit-flips produced in flip-flops, everything in a single solution.

3 Multilevel Emulation-Based Fault Injection Approach

A SET can be modeled as a spurious voltage pulse on the output of a gate. The pulse may propagate across the circuit and eventually provoke malfunction. The purpose of fault injection is to inject SETs in a circuit and classify their effects. The resulting classification is an estimation of SET sensitivity for the circuit under test. In order to estimate the SER before the circuit is manufactured, fault injection is performed on the model of the circuit under test that results from the design process.

In emulation-based fault injection, the model of the circuit is downloaded into a FPGA. The voltage pulse induced by a SET is modeled at the logic level as an erroneous logic value (0 or 1) at the output of a logic gate that lasts for the duration of the pulse. Propagation of the pulse is then performed by executing the circuit in the FPGA and the fault effect is classified by comparing the result with a golden execution. Therefore, the emulation system must support fault injection at any gate and time instant, comparison between the golden and faulty execution, classification of fault effects, and external communication with the user. In Autonomous Emulation, all these functions are implemented inside the FPGA in order to improve emulation efficiency.

Propagation of a SET effect can be seen as a two-step process [13]. First the pulse is propagated throughout the combinational logic up to the memory elements (latches, flip-flops and memories). At this point, the SET effect can be seen as an SEU, if just one memory element or bit is affected, or as an Multiple Bit Upset, in case several memory elements or bits are affected. If the SET does not produce any change on the circuit state, it can be classified as having no effect. Otherwise, if the SET produces a SEU or MBU, then a second stage is needed. In the second stage, the SEU/MBU is propagated in subsequent clock cycles until the fault effect is finally classified.

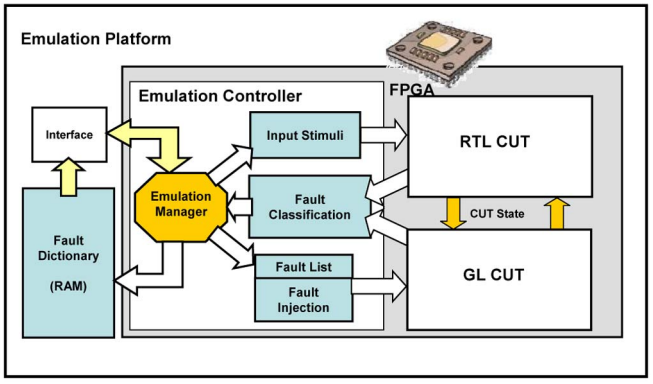

Fig. 2. General architecture of the AMUSE system.

Emulation of SET effects must be done at least at the gate level. In particular, the gate level model must include delay information of the circuit for the target technology, because the propagation of pulses across combinational logic is determined by delays along with the logic functionality. This makes the emulation process rather slow, in comparison with SEU emulation, which can be made at the register-transfer level. However, once the transient pulses have reached the memory elements, emulation can be made at the register-transfer level for a synchronous circuit.

Based on these considerations, we propose a multilevel instrumented emulation approach. Fig. 2 shows a general diagram of the multilevel emulation system. In this approach, two models of the CUT are built at the GL and RTL, respectively. The two models can interact by passing the current state of the circuit in both directions, from the RTL model to the GL model and vice versa. The AMUSE emulation system can switch between these models at any time during emulation.

The multilevel emulation system contains an emulation controller along with models of the microprocessor under test at the GL and the RTL. The emulation controller is a general-purpose configurable module that provides input stimuli for the CUT, activates fault injection in the GL model, coordinates execution and performs the classification of the faults. The emulation manager also communicates with a host through an external interface in order to configure the fault injection campaign and to send the results. Partial results can be stored temporarily in external memory in order to make more efficient the communication with the host.

In the following sections, the main components of the multilevel emulation system are described.

4 The Gate-Level Instrumented Model

The GL instrumented CUT is a functionally equivalent model of the CUT that includes delays and supports fault injection and emulation at gate level. The GL instrumented CUT is obtained by substituting every component in the CUT netlist by its equivalent instrumented component.

A crucial aspect is the implementation of arbitrary circuit delays into an FPGA. This aspect will be described in the following section. Then, the instrumented cells are described in Section 4.2.

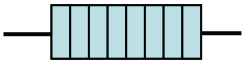

Fig. 3. Delay element modeled as a shift register (TQ model).

4.1 Embedding Delays in the FPGA Emulation Model

The propagation of a pulse across combinational logic is delay-dependent. Therefore, in order to analyze SET effects by FPGA emulation, the delay information of the CUT must be embedded in the FPGA model.

Delays can be implemented in FPGAs by means of Time Quantization (TQ) [14]. In this approach, each delay is rounded to an integer multiple of a selected time quantum. Then, delays are implemented as shift registers driven by a time quantization clock (Fig. 3). The number of time quanta of each delay determines the length of the corresponding shift register. A shift register can be implemented in a FPGA very efficiently with just one LUT.

The time quantization approach allows embedding arbitrary delays in a FPGA. Time advance on a time quantum basis is performed by a time quantization clock. This must not be confused with the system clock. The system clock period is also quantized, so that one system clock period generally consists of many time quanta.

The TQ model is a basic delay model. In particular, it does not cover inertial delay or delay degradation effects, which cause the electrical masking effect. To cover for these effects, a Voltage-Time Quantization (VTQ) approach has been proposed in [15]. The VTQ approach considers the quantization of the voltage-time transition curves. A quantized rising/falling transition curve is executed by a nonlinear counter (Fig. 4) driven by a time quantization clock. The quantized transition curves are tabulated and the nonlinear counter follows the rising or falling curve depending on the logic value of the input. The VTQ model can implement partial transitions that are produced when the input switches very fast. Inertial delay and delay degradation effects, which are caused by partial transitions, are well covered in this model. The reader is referred to [15] for details regarding the accuracy of the VTQ model.

4.2 Equivalent Instrumented Cells at Gate Level

In order to obtain the instrumented GL model, each cell in the original circuit netlist is replaced by its equivalent instrumented cell. The GL instrumented cells are functionally equivalent and implement delays according to the models described in the previous section. In addition, they include additional inputs and outputs to support fault injection and communication with the RTL model.

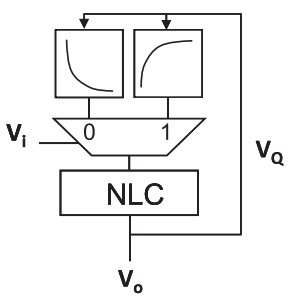

Fig. 4. Delay element using a Voltage-Time Quantization model.

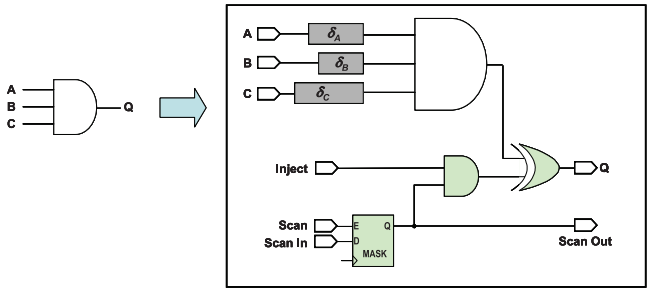

Fig. 5. GL Instrumented gate.

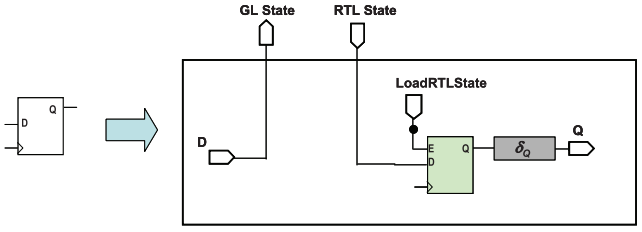

Fig. 6. GL Instrumented FF.

Figs. 5 and 6 show the GL instrumented model for a gate and a FF, respectively. In the case of a gate, the instrumented model includes a parameterizable delay element at each input, that accounts for net and gate delay. The fault injection logic is implemented by the mask FF and associated gates. The mask FFs of all gates form a scan path that allows to select the fault injection target by introducing serially a pattern in the scan path. At fault injection time, a high level pulse is applied to the inject signal which changes the output of the gate(s) selected by the mask.

In the case of a FF, the input and output of the FF are decoupled in the instrumented model (Fig. 6). The data input (D) is redirected to the RTL model and the input of the instrumented FF is taken from the RTL model when the LoadRTLState signal is asserted. This is required in order to avoid passing a faulty state to the output of the FF. Instead, the faulty data that may be produced as a result of fault injection at the gate level is sent to the RTL via the GL State signal in order to proceed with emulation at the RTL. Conversely, the LoadRTLState signal enables loading the RTL state coming from the RTL model. The LoadRTLState signal is activated in the clock cycle where fault injection at GL is to take place.

When the LoadRTLState signal is asserted, each instrumented FF loads the RTL state from the RTL model in the FF on the right of Fig. 6. Then, the logic values at the FF outputs (Q) propagate through the logic gates. At injection time, an active pulse is set on the inject signal which produces a faulty pulse on the output of the gate or gates whose mask signal is asserted. Then propagation continues. At the end of the current clock cycle, the D signals of all FFs contain the propagated values. Finally, the GL State is uploaded in the RTL CUT in order to propagate the fault effect across subsequent clock cycles.

The instrumented model for a memory is similar to the instrumented FF model. The input data, address and control signals are passed to the RTL and decoupled from the output of the cell. The memory is actually implemented at the RTL. Memory output data are loaded from the RTL model when the LoadRTLState signal is activated.

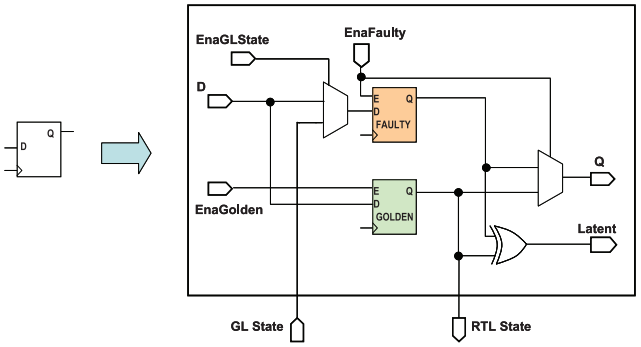

Fig. 7. RTL Instrumented FF.

The GL model supports injection of SETs and SEUs. SEU fault injection is accomplished by injecting a pulse at the input of the FF just before the GL State is uploaded in the RTL model.

5 The Register-Transfer Level Instrumented Model

The RTL instrumented CUT is a functionally equivalent model of the circuit under test that supports fault emulation at RT level. At this level, the time resolution is one clock cycle and propagation delays can be abstracted as long as they satisfy the clock constraints.

In order to obtain the instrumented RTL model, each cell in the original circuit netlist is replaced by its equivalent instrumented cell. The instrumented cells are functionally equivalent and implement the comparison of faulty and golden execution in order to support the classification of fault effects. They also include additional inputs and outputs to support communication with the GL model.

The instrumented RTL model for a gate is just the functional equivalent of the original gate, without any instrumentation.

The instrumented RTL model for a FF is shown in Fig. 7. It contains two FFs, called Faulty FF and Golden FF, that are used to store the Faulty and Golden states, respectively. A multiplexer is added at the input of the Faulty FF in order to select between the faulty state provided by the GL model or the RTL model. The Latent output indicates the discrepancy between the Faulty and Golden states.

The RTL model is used first to reach the clock cycle at which fault must be injected. During this phase the Golden and Faulty FFs store the same state. Then, the current state of the circuit is passed to the GL model via the RTL State signal. When fault injection is completed at the GL, the EnaGLState signal is activated and the GL state is uploaded into the Faulty FF. At this moment, the golden and faulty states are compared. If there are no differences, we can conclude that the fault effect has not been propagated to any FF. Otherwise, execution continues at the RTL, comparing the golden and faulty FF at each clock cycle, until a classification is achieved.

Execution is performed using a time-multiplexed approach [9], where faulty and golden execution take place at alternate clock cycles. To this purpose, the emulation

manager alternates the activation of the EnaGolden and EnaFaulty signals during the fault propagation phase. This way the combinational logic is shared and there is no need to replicate it.

Note that the RTL model does not need to support fault injection, as this task is implemented at the gate level. The initial faulty state is uploaded from the GL model and then propagated in the RTL model.

The RTL model for a memory is also similar to the instrumented FF model. It contains two memories, one for the faulty contents and another one for the golden contents. The input data, address and control signals are taken from the GL model and passed to the faulty memory when the GL model execution phase completes, or from the RTL signals during the RTL execution phase. Memory output data are taken from the faulty memory or the golden memory as selected by the EnaFaulty signal.

The generation of the Latent signal, which is the result of comparing the faulty and golden states, is far more complex in the case of a memory. The Latent signal must be active whenever there is at least one difference between the contents of the faulty memory and the golden memory. There is also the possibility that an error is canceled in the faulty memory. To satisfy these requirements, the memory model contains a small Content Address Memory (CAM) that keeps track of errors. The CAM stores a tag for each memory position where the Faulty and Golden memory differ. The CAM is checked and updated at each memory access.

6 The Emulation Controller

The Emulation Controller coordinates the overall emulation process. Consider a fault to be injected at time Ti with a duration D. Ti can be expressed as

$$\mathrm{Ti} = \mathrm{N} \times \mathrm{Tclk} + \mathrm{ti},$$

where N is the number of clock cycles from the initial emulation state and ti is the injection time referred to the clock cycle where injection takes place. In the GL model, ti and D are further rounded to integer multiples of the time quantum.

Emulation of a SET is performed with the following steps:

1. Reset the RTL model.
2. Advance emulation up to clock cycle N in order to set the circuit to the state immediately before SET fault injection. This step is performed in the RTL circuit.
3. Transfer the state from the RTL circuit to the GL circuit.
4. Advance emulation in the GL circuit up to the injection time ti. Inject pulse at time ti and keep the pulse for the duration D. Then continue up to the end of the current clock cycle.
5. Transfer the resulting faulty state from the GL circuit to the RTL circuit. The Faulty FFs in the RTL circuit contain the faulty state at clock cycle $\mathrm{N}+1$.
6. Advance golden emulation in the RTL circuit by one clock cycle. The Golden FFs in the RTL circuit contains the golden state at clock cycle $\mathrm{N}+1$.
7. Compare golden and faulty states for all FFs and memories. If there are no differences, the fault is classified as having no effect. Otherwise, advance emulation in the RTL circuit until the fault can be classified or the end of the emulation is reached.
8. Fault is classified. Go to step 1 for next fault.

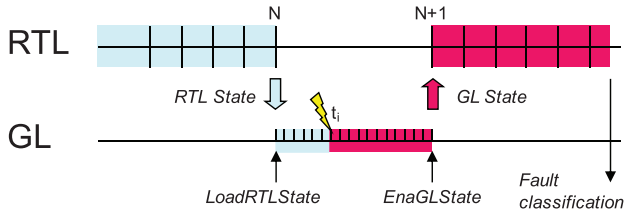

Fig. 8. Emulation execution.

For the classification of the fault effects, we consider at least three fault categories:

- Silent. The pulse effects eventually disappear. This is detected when the golden and faulty states become identical for all FFs and memories after fault injection.
- Failure. The pulse effects are propagated to at least an output of the circuit under test.
- Latent. The pulse effects were not propagated to an output of the circuit, but the internal state of the circuit is not equivalent at the end of the execution.

Fig. 8 illustrates emulation execution. The GL and the RTL circuits work at different times. The GL circuit is used around injection time, when a fine time resolution is needed. Before injection time, or after the SET has propagated to some memory element, clock cycle resolution is enough. Then, the RTL model is used to speed up emulation until fault classification is achieved.

Note that the RTL and GL execution are controlled by enable signals rather than using different clocks. The whole model is synchronous, but an FPGA clock tick corresponds to a different amount of time advance in each model. In the GL model it is a time quantum, while in the RTL model it is a system clock period. Therefore, the RTL model speeds up execution by a factor equal to the number of time quanta in a clock cycle.

7 Construction of the Emulation System

The construction of the AMUSE emulation system for a given circuit is done automatically with the use of the tools developed by the authors. Fig. 9 illustrates this process.

The starting point is a netlist of the CUT and its associated delays in Standard Delay Format (SDF). These files are produced by the synthesis process and are the same files that are used for the postsynthesis or postlayout simulation. In the first step, an instrumentation tool is used to generate the instrumented RTL and GL models of the CUT. The instrumenter substitutes every cell in the original netlist by its equivalent cell in the RTL and GL libraries, respectively. The instrumenter also connects appropriately the extra control and data external signals used by the instruments.

The RTL and GL libraries are generic cell libraries that contain synthesizable instrumented models of the cells, as described in the previous sections. These libraries cover most of the cells that can be found in any digital cell library,

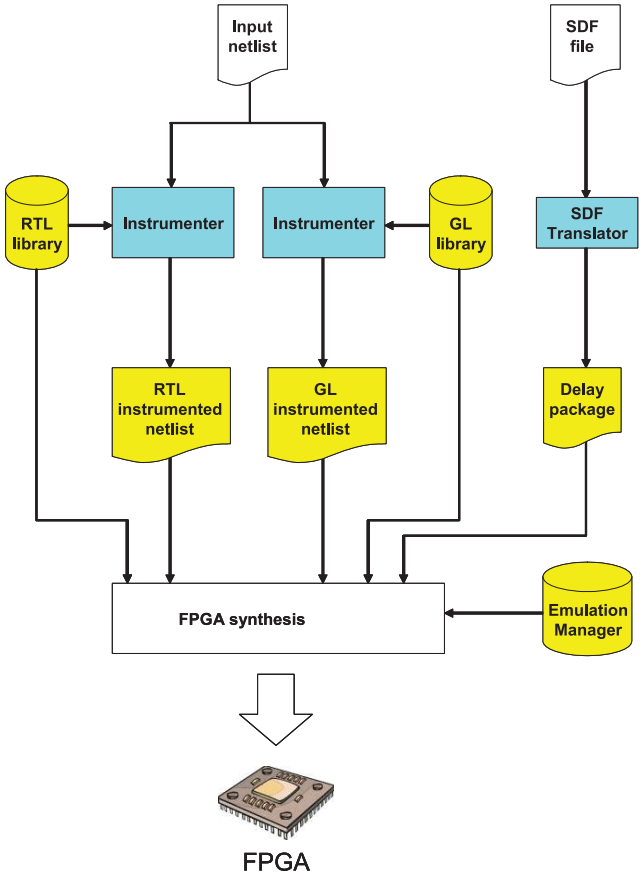

Fig. 9. Construction of the emulation system.

including complex gates, different types of FFs and memory blocks. The libraries have a hierarchical and configurable structure, so that changing the delay model is immediate. Obviously, the libraries must correspond to the particular technology library used for the design. Mapping a particular technology library to the generic library is straightforward by just associating cell names. Also, adding new cells is simple, because only the logic function must be specified and the instruments and delay models are hierarchically inherited.

The GL cells include delays as generic parameters. For delay backannotation, the SDF file is translated automatically into a VHDL package of constants. The instrumenter maps each of these constants to the generic delay parameters of each cell.

Once the instrumented models are generated, the complete system consists of the following components: the instrumented RTL netlist, the RTL library, the instrumented GL netlist, the GL library, the delay package, and the emulation controller module. The system is compiled and loaded into the FPGA using the conventional toolset provided by the FPGA vendor. The system can be compiled into any FPGA of any brand, except for the interface to the host and to external on-board resources.

8 Experimental Results

An AMUSE system has been built around the LEON2 processor. LEON2 is a 32-bit processor conforming to the SPARC V8 architecture, well suited for embedded applications [19]. It has separate instruction and data caches, interrupt controller, two 24-bit timers, two UARTs, 16-bit I/O port, a flexible memory controller, and a full implementation of AMBA AHB and APB on-chip buses. The integer unit has a 5-stage instruction pipeline and support for up to 32 register windows.

The LEON2 VHDL model IP can be configured in several ways. For the experiments, the selected configuration includes:

- Eight register windows.
- 2 kB instruction cache.
- 2 kB data cache.
- 256 kB on-chip SRAM memory.

The synthesis results with the LEON2 case study are presented in the following section. Then, fault injection results are given in Section 8.2.

TABLE 1
Original Netlist Data

ASIC resources	Gates	FFs	Mem. (Kb)
Original netlist	8,310	1,466	2,090.5
FPGA resources	LUTs	FFs	BRAMs
Logic	4,343	1,518	
Memories	298	304	70
TOTAL	4,641	1,822	70

8.1 Synthesis Results

Table 1 shows the data for the original netlist of the LEON2 microprocessor. The total number of cells is 9,783, more specifically 8,310 combinational cells, 1,466 flip-flops, and seven memory cells. The memory cells are the register file, the caches, and the SRAM memory.

For the sake of comparison, we have also included in Table 1 the results obtained when the original netlist is synthesized for the FPGA used in the experiments. However, it must be noted that this is just to evaluate the efficiency of FPGA synthesis for the CUT. The fault injection experiments are performed with the original netlist, which is an ASIC netlist. The results are separated for logic and memories, in order to show the contribution of each of them. The number of LUTs is smaller than the number of gates, as a LUT can map more than one gate. The number of flip-flops is slightly higher than in the original netlist, because the FPGA synthesis tool duplicates some flip-flops to improve timing.

Two emulation systems, using the TQ and the VTQ delay models, respectively, have been generated from the original netlist, synthesized with Synplify Pro and Xilinx ISE 11.5 and implemented in a Xilinx XC6VLX240T FPGA [18]. The synthesis results in terms of LUTs, FFs, and Block RAMs (BRAMs) are shown in Tables 2 and 3, respectively. Each of these tables shows the sizes of the RTL model of the CUT, the instrumented memories, the GL model of the CUT, the emulation controller, the complete emulation system, and the percentage of FPGA usage. The construction of a complete emulation system from the original netlist takes about 1 hour and a half in a PC, including instrumentation and FPGA synthesis.

TABLE 2
Synthesis Results with TQ Delay Model

Component	LUTs	FFs	BRAMs
RTL CUT	9,551	2,962	
Memories	2,501	2,763	140
GL CUT	26,564	9,890	
Emulation Controller	1,451	1,095	
TOTAL	40,067	16,710	140
TOTAL (%)	26.58%	5.54%	33.65%

TABLE 3
Synthesis Results with VTQ Delay Model

Component	LUTs	FFs	BRAMs
RTL CUT	9,551	2,962	
Memories	2,501	2,763	140
GL CUT	49,541	58,648	
Emulation Controller	1,451	1,095	
TOTAL	63,044	65,468	140
TOTAL (%)	41.83%	21.72%	33.65%

For both models, the time quantum was selected in such a way that the maximum single delay fits in a 16 bit shift register or in 4 bit nonlinear counter. This results in a total of 71 time quanta at the maximum clock frequency of the circuit.

The results in Tables 2 and 3 show that the overhead of including the RTL model along with the GL model is just 24 percent of LUTs and 18 percent of FFs in the TQ case, and 15 percent of LUTs and 5 percent of FFs, in the VTQ case. This is an acceptable overhead, considering the benefits that can be obtained in terms of speed up. Since the RTL CUT executes one system clock cycle in one FPGA clock cycle, the speed-up factor with respect to pure GL emulation can be roughly estimated as the number of time quanta in a clock cycle, which typically is about two orders of magnitude.

Instrumented memory models introduce a significant overhead, due to the need to implement a CAM for each memory cell to keep track of errors. This overhead depends on the size of memory buses. The resources used by the instrumented memories is shown in a separate row in Tables 2 and 3. The number of Block RAMs is twice the noninstrumented implementation, because of the need to implement the golden and the faulty memories.

The implementation of the TQ delay model requires a large number of LUTs to implement the shift registers. However, with the VTQ model the number of LUTs and FFs is balanced. The VTQ model requires 57 percent more LUTs and 3.9 times of flip-flops than the TQ model, but it is more accurate.

Both systems fit comfortably in a XC6VLX240T FPGA, which is available in general-purpose evaluation boards, such as the ML605 Evaluation Kit [18]. According to Tables 2 and 3, the critical resource is the number of LUTs. Note that the VTQ model requires more flip-flops in absolute terms, but the percentage of use is smaller because Virtex 6 FPGAs have two flip-flops per LUT. Discounting the Emulation Manager, which is roughly the same for all circuits, the average ratio of LUTs per gate is 4.6 for the TQ model and 7.4 for the VTQ model. Considering the resource usage for this example and the size of the largest available Virtex 6 FPGA, namely the XC6VLX760 with 474,240 LUTs [18], the AMUSE system can be implemented in a single FPGA for circuits up to 80 K gates with the TQ model and up to 50 K gates with the VTQ model, assuming 80 percent FPGA usage. Larger circuits can be implemented with multiple FPGAs. Alternatively, a partial implementation of the GL model for just one block of the CUT can be used to save resources.

With respect to memories, the XC6VLX760 has 25,920 Kb, so the AMUSE system can implement on-chip up to 12,960 Kb of memory. External memory can be used to implement larger memories.

TABLE 4
Fault Injection Results

Pulse width	Tclk	0.1 Tclk	0.1 Tclk
Delay model	TQ/VTQ	TQ	VTQ
Workload (clock cycles)	20,000	20,000	20,000
# Faults injected	164,720,820	59,308,470	59,308,470
# Silent	115,625,249	57,594,453	57,613,554
# Failure	27,779,764	392,879	377,992
# Latent	21,148,887	1,319,353	1,315,943
# RAM Latent Only	166,920	1,785	981
Runtime (sec.)	52,520	15,735	15,733
FI rate (faults/sec.)	3,136	3,769	3,770

8.2 Fault Injection Results

Several fault injection experiments were performed in order to evaluate the capabilities and the performance of the AMUSE system. For the experiments, the LEON2 was running a simple Bubble Sort algorithm and the results were sent to on-chip memory. The outputs of the circuit are the memory bus signals, including address, data, write enable, and chip select signals. The experiments were conducted on a Xilinx ML605 board [18].

Table 4 summarizes the results of the experiments. For each experiment we report the pulse width as a fraction of the system clock period, the delay model used, the workload length, the number of silent, failure, latent, and RAM latent faults, the total time required for the experiment and the average fault injection rate. The Failure count refers to faults that produce errors at the outputs or a number of errors stored in a memory that exceeds the size of the CAM. In the experiments, the size of the CAM is set to four positions. The Latent count refers to faults that are not classified as failure, but at least one flip-flop stores a wrong value at the end of the execution. The RAM Latent Only count refers to faults that do not produce errors at the outputs nor at the flip-flops, but memory contents are corrupted without exceeding the size of the CAM.

In the first experiment, fault injection was executed with a pulse length equal to the system clock period. Thus, in this case, no timing masking is possible and only logic masking is taken into account. Faults were injected for all signals in every clock cycle. About 16 percent of the faults are classified as failure and 13 percent of the faults as latent. These figures represent the worst case behavior of the circuit.

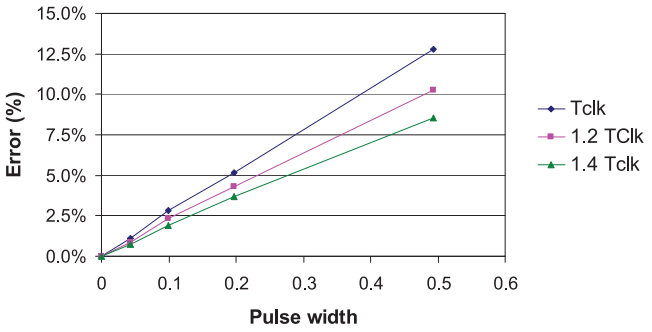

Fig. 10. Sensitivity versus pulse width for several clock periods.

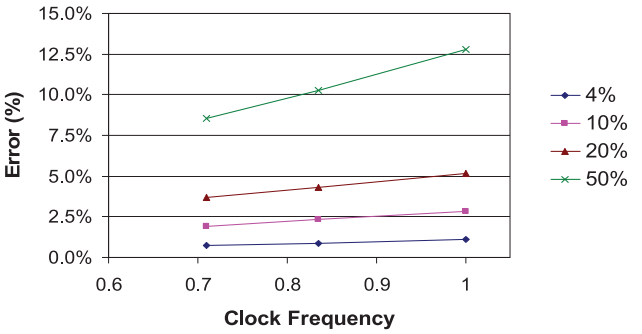

Fig. 11. Sensitivity versus clock frequency for several pulse widths.

The next columns in Table 4 show the results for a fault injection campaign using a pulse width of 10 percent of the system clock cycle. These results were obtained by randomly injecting faults at any clock cycle and at any time quantum inside the clock cycle. In these cases, the number of failure and latent faults reduce, respectively, to 0.6 and 2.2 percent of the total number of faults. The number of failure and latent faults with the VTQ model is slightly smaller than with the TQ model due to electrical masking effects.

Although the figures are low in the latter case, it must be noted that these are referred to the whole area of the circuit. In comparison, SEU figures are typically higher but they are just referred to the area of the circuit occupied by flip-flops and the probability of a single event is smaller. This fact emphasizes the need for large fault injection campaigns in order to obtain significant results.

In all cases, the fault injection rate is in the order of 3,000 faults per second running at a moderate frequency of 40 MHz. Therefore, the AMUSE system can perform massive fault injection campaigns, in the order of tens of millions of faults within acceptable time. In comparison, the closest approach that performs SET fault injection and propagation across clock cycles [13] reports a fault injection rate in the order of about one fault per second.

In a second set of experiments, we measured the error rate for several pulse widths and system clock periods. The results are shown graphically in Figs. 10 and 11 for the VTQ model. The experiments were also made for the TQ model, but the results are very similar and the figures are omitted because the differences cannot be appreciated graphically.

The curves in Fig. 10 show the error percentage as a function of the pulse width measured as a fraction of the minimum clock period (Tclk). Three curves have been obtained, namely for Tclk, 1.2 Tclk, and 1.4 Tclk.

The curves in Fig. 11 show the error percentage as a function of clock frequency for several pulse widths. The clock frequency scale is relative to the maximum clock frequency. As expected, the sensitivity increases with the clock frequency and the pulse width.

9 Conclusions

SET sensitivity assessment by fault injection requires large fault injection campaigns that must be performed at least at the gate level to take delays into account. In order to increase performance, the AMUSE emulation system integrates seamlessly the RTL model and the gate level model of the circuit under test in a FPGA. Switching between these models can be made at any time during emulation. Thus, fault injection is performed accurately at the gate level, while fault propagation across clock cycles is performed at the RTL for faster execution.

The overhead produced by introducing the RTL model is largely compensated by the performance benefits, since the RTL model executes two orders of magnitude faster than the GL model. With this approach, large fault injection campaigns are feasible within acceptable time. Complex circuits, including microprocessors, memories, and peripherals fit in currently available FPGAs.

Acknowledgments

This work was supported in part by the Spanish Ministry of Science and Innovation under project ESP2007-65914-C03-01.

References

[1] R.C. Baumann, "Radiation-Induced Soft Errors in Advanced Semiconductor Technologies," *IEEE Trans. Device and Materials Reliability,* vol. 5, no 3, pp. 305-316, Sept. 2005.

[2] M. Nicolaidis, *On-Line Testing for VLSI.* Springer, 1998.

[3] P. Bernardi, L.M. Veiras Bolzani, M. Rebaudengo, M. Sonza Reorda, F.L. Vargas, and M. Violante, "A New Hybrid Fault Detection Technique for System-on-a-Chip," *IEEE Trans. Computer,* vol. 55, no. 2, pp. 185-198, Feb. 2006.

[4] L. Berrojo, I. González, F. Corno, M. Sonza Reorda, G. Squillero, L. Entrena, and C. López Ongil, "An Industrial Environment for High-Level Fault-Tolerant Structures Insertion and Validation," *Proc. 20th IEEE VLSI Test Symp.,* pp. 229-236, 2002.

[5] M.-C. Hsueh, T.K. Tsai, and R.K. Iyer, "Fault Injection Techniques and Tools," *IEEE Computer,* vol. 30, no. 4, pp. 75-82, Apr. 1997.

[6] L. Antoni, R. Leveugle, and B. Feher, "Using Run-Time Reconfiguration for Fault Injection in HW Prototypes," *IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems,* pp. 245-253, 2002.

[7] M.A. Aguirre, J.N. Tombs, V. Baena, F. Muñoz-Chavero, A. Torralba, A. Fernández-León, and F. Tortosa, "FT-UNSHADES: A New System for SEU Injection, Analysis, and Diagnostics over Post Synthesis Netlist," *MAPLD NASA Military and Aerospace Programmable Logic Devices,* 2005.

[8] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, and M. Violante, "Exploiting Circuit Emulation for Fast Hardness Evaluation," *IEEE Trans. Nuclear Science,* vol. 48, no 6, pp. 2210-2216, Dec. 2001.

[9] C. López-Ongil, M. García-Valderas, M. Portela-García, and L. Entrena, "Autonomous Fault Emulation: A New FPGA-Based Acceleration System for Hardness Evaluation," *IEEE Trans. Nuclear Science,* vol. 54, no. 1, pp. 252-261, Feb. 2007.

[10] M. Nicolaidis, "Emulation/Simulation d'un Circuit Logique," French patent, filed Feb. 25, 2005, issued, Oct. 2007.

[11] M. García Valderas, P. Peronnard, C. López Ongil, R. Ecoffet, F. Bezerra, and R. Velazco, "Two Complementary Approaches for Studying the Effects of SEUs on Digital Processors," *IEEE Trans. Nuclear Science,* vol. 54, no. 4, pp. 924-928, June 2007.
[12] M. Violante, "Accurate Single-Event-Transient Analysis via Zero-Delay Logic Simulation," *IEEE Trans. Nuclear Science,* vol. 50, no. 6, pp. 2113-2118, Dec. 2003.
[13] M.A. Aguirre, V. Baena, J. Tombs, and M. Violante, "A New Approach to Estimate the Effect of Single Event Transients in Complex Circuits," *IEEE Trans. Nuclear Science,* vol. 54, no. 4, pp. 1018-1024, Aug. 2007.
[14] M. García Valderas, R. Fernández Cardenal, C. López Ongil, M. Portela García, and L. Entrena, "SET Emulation under a Quantized Delay Model," *J. Electronic Testing: Theory and Applications,* vol. 25, no. 1, pp. 107-116, 2009.
[15] L. Entrena, M. García-Valderas, R. Fernández-Cardenal, M. Portela-García, and C. López-Ongil, "SET Emulation Considering Electrical Masking Effects," *IEEE Trans. Nuclear Science,* vol. 56, no. 4, pp. 2021-2025, Aug. 2009.
[16] F. Wang and Y. Xie, "An Accurate and Efficient Model of Electrical Masking Effect for Soft Errors in Combinational Logic," *Proc. Second Workshop System Effects of Logic Soft Error (SELSE2),* Apr. 2006.
[17] D. Alexandrescu, L. Anghel, and M. Nicolaidis, "Simulating Single Event Transients in VDSM ICs for Ground Level Radiation," *J. Electronic Testing: Theory and Applications,* vol. 20, no. 4, pp. 413-421, 2004.
[18] http://www.xilinx.com, 2011.
[19] J. Gaisler, "The LEON Processor User's Manual," *Gaisler Research,* Aug. 2001.
[20] V. Ferlet-Cavrois, P. Paillet, D. McMorrow, N. Fel, J. Baggio, S. Girard, O. Duhamel, J.S. Melinger, M. Gaillardin, J.R. Schwank, P.E. Dodd, M.R. Shaneyfelt, and J.A. Felix, "New Insights into Single Event Transient Propagation in Chains of Inverters-Evidence for Propagation-Induced Pulse Broadening," *IEEE Trans. Nuclear Science,* vol. 54, no. 6, pp. 2338-2346, Dec. 2007.
[21] V. Ferlet Cavrois, V. Pouget, D. McMorrow, J.R. Schwank, N. Fel, F. Essely, R.S. Flores, P. Paillet, M. Gaillardin, D. Kobayashi, J.S. Melinger, O. Duhamel, P.E. Dodd, and M.R. Shaneyfelt, "Investigation of the Propagation Induced Pulse Broadening (PIPB) Effect on Single Event Transients," *IEEE Trans. Nuclear Science,* vol. 55, no. 6, pp. 2842-2853, Dec. 2008.
[22] A. Benso, M. Rebaudengo, L. Impagliazzo, and P. Marmo, "Fault-List Collapsing for Fault-Injection Experiments," *Proc. Ann. Reliability and Maintainability Symp.,* pp. 383-388, 1998.

Luis Entrena received the MS degree from the Universidad de Valladolid, Spain, in 1998, and the PhD degree from the Universidad Politécnica de Madrid, Spain, in 1995. From 1990 to 1993, he was with AT&T Microelectronics, New Jersey. From 1993 to 1996, he was with TGI, Spain. Since 1996, he is an associate professor at Universidad Carlos III de Madrid, Spain, where he has been head of the Electronic Technology Department. His current research interests include online testing, fault tolerance, logic synthesis, and hardware acceleration.

Mario García-Valderas received the MS degree in industrial engineering in 1997 and the PhD degree in digital electronics in 2004, both from the Universidad Politécnica de Madrid, Spain. Currently, he is an assistant professor in the Electronic Technology Department at Universidad Carlos III de Madrid, Spain. His research interests include fault tolerant system design and evaluation, hardware acceleration using FPGAs, and digital design techniques and tools.

Raul Fernández-Cardenal received the industrial engineer degree from Universidad Carlos III de Madrid, Spain, in 2005. He is currently working toward the PhD degree in the Electronic Technology Department at Universidad Carlos III de Madrid. He is also working for INDRA SA company in the Euroradar CAPTOR program. His research interests include design, validation, and test of fault tolerant electronic systems with particular emphasis in Single Event Transient fault effects.

Almudena Lindoso received the MS degree in 2001 from the University Politécnica of Madrid and the PhD degree in 2009 from the University Carlos III of Madrid. Since 2003, she has been an assistant professor and a researcher in the Electronic Technology Department at University Carlos III of Madrid. Her current research interests include image processing, fault tolerance, hardware acceleration, and reconfigurable devices.

Marta Portela García received the PhD degree in electronic engineering from the University Carlos III of Madrid in 2007. Currently, she is an assistant professor in the Electronic Technology Department of the University Carlos III of Madrid, Spain. Her research interests mainly include fault tolerance evaluation in digital circuits, transient fault emulation using FPGAs, SoC testing, and VLSI circuits design.

Celia López-Ongil received the MS degree in industrial engineering in 1995 and the PhD degree in digital electronics in 2000, both from the Universidad Politécnica de Madrid, Spain. She has been working in Universidad Carlos III de Madrid since 1998, where now she is an associate professor in the Electronic Technology Department. Her research interests include design techniques for reliable digital integrated circuits, CAD tools for enhancing the design process, and fault tolerance.