Microelectronics Reliability 69 (2017) 115–125

Contents lists available at ScienceDirect

Microelectronics Reliability

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

Emulation-based fault analysis on RFID tags for robustness and security evaluation

Ibrahim Mezzah [a,b,*], Hamimi Chemali [b], Omar Kermia [a]

[a] Centre de Développement des Technologies Avancées (CDTA), Algiers, Algeria
[b] Department of Electronics, University of Setif, Setif, Algeria

ARTICLE INFO

Article history:
Received 10 August 2016
Received in revised form 15 December 2016
Accepted 16 December 2016
Available online 20 December 2016

Keywords:
RFID tag
Fault injection
SEU emulation
FPGA platform
Robustness

ABSTRACT

This paper presents an FPGA (field-programmable gate array) based fault emulation system for analysis of fault impact on security and robustness of RFID (radio frequency identification) tags. This emulation system that deals with any RFID protocol consists of two tag-reader pairs, a fault injection module and an emulation controller all implemented in a single FPGA. The designed approach performs single event upset (SEU) and single event transient (SET) fault injection and permits with high flexibility to set communication scenarios and related parameters. Moreover, we propose a classification of produced errors to evaluate fault impacts and identify most sensitive tag flip-flops causing large number of failures and security concerns. The proposed fault injection approach provides suitable means to increase tags' security and robustness. In our experimentation campaign, an ultra-high frequency (UHF) tag architecture has been exposed to intensive SEU and SET fault injections. The duration of the campaign including results analysis is 30 min in where 6,215,316 faults are experimented. Our results have shown that the tag has tolerated 61.82% of SEUs and 67.83% of SETs. The flip-flops that constitute the tag FSM (finite state machine) have been identified as the most sensitive parts causing large number of failures.

1. Introduction

Advances in radio frequency identification (RFID) technology have covered a large variety of modern applications fields and recently extra interests are focused on national security and the internet of things. However, deployment of this technology is facing many challenges concerning authentication, security and reliability. Generally, an RFID system consists of *readers* and *tags* communicating through radio waves. The tag (transponder) consists of a small chip and an antenna; the reader (interrogator) has the charge of monitoring tag data. Since tags are electronic components, so they are vulnerable to external disturbance and susceptible to fault attacks [1]. Thus, the evaluation of tag fault tolerance and its security assessments are compulsory. Fault analysis provides evaluation tools of tag robustness and identifies highly vulnerable tag parts.

Single event upset (SEU) and single event transient (SET) represent potential faults occurring in electronic devices resulting from heavy particle effect. SEU fault is a change in state of a storage element, while SET occurs in combinational cells as a transient output pulse. More interest in these types of faults is solely resulting from semiconductor technology progress, involving smaller transistors and reduced power supply. Currently, estimating SEU and SET sensitivity is a crucial task in safety-critical circuit design [2].

The purpose of this work is to carry out an SEU and SET effect analysis on RFID tags for evaluating robustness, identifying errors having an impact on tag security and revealing tag weak parts. Indeed, more latitude of changes with less compromising is possible in earlier design steps. Security improvement should result through enhancements and perfection of the tag design or via appropriate countermeasure actions on targeted weak parts.

Fault injection techniques are widely used to carry out fault sensitivity analysis for integrated circuits. Simulation-based fault injection approach is then frequently used to estimate the circuit sensitivity at design level [3,4]. This technique provides high flexibility since it supports several fault models and configurable fault campaigns but requires a massive computational effort. However, emulation-based fault injection is still considered as unavoidable alternative means for accelerating fault injection experiments. This technique uses field-programmable gate array (FPGA) devices to emulate the circuit under test behaviour [5–7]. In RFID, when considering a communication between a reader and tags during an inventory round, infinity of communication scenarios, from the point of view of commands and responses progress, is possible. In fact, for any given scenario, several parameters can be supported including reader-tag calibration (RTcal), tag-reader calibration (TRcal) and communication encoding type from tags to the reader (FM0 or Miller encoding). Thus, emulation-based fault injection provides valuable means for performing extensive fault injection

* Corresponding author at: Centre de Développement des Technologies Avancées (CDTA), Algiers, Algeria.
E-mail addresses: imezzah@cdta.dz (I. Mezzah), okermia@cdta.dz (O. Kermia).

http://dx.doi.org/10.1016/j.microrel.2016.12.008
0026-2714/

experimentations covering multiple scenarios implying several sets of parameters.

In this paper, we have developed a new FPGA-based fault emulation platform for RFID systems. This platform is used to perform SEU and SET analysis on an ultra-high frequency (UHF) RFID tag compatible with electronic product code class-1 generation-2 (EPC Gen2) standard [8]. However, this approach is also applicable to other RFID protocols. It should be noticed that the purpose of emulation-based fault injection is for evaluating the SEU sensitivity of an ASIC (application-specific integrated circuit) design, not an FPGA design. SEU effects appear in the memory elements of circuits, which are the same in the ASIC and in the FPGA prototype [7,9]. As far as SET experimentations are concerned, we considered SETs that affect tag flip-flops outputs and preserve valuable results for ASIC design. In addition, we considered that applied SETs have sufficient duration that affects the circuit without taking into account complicated aspects accompanying SET fault as detailed and discussed in several works [10,11].

The paper is organised as follows. Section 2 reviews RFID tag fault analysis and fault injection practice related works. Then, the developed fault emulation system is presented and detailed in Section 3. Section 4 presents the adopted fault emulation process. Fault experimentation and obtained results are discussed in Section 5 and Section 6 respectively. Finally, Section 7 states the conclusions of this work.

2. Related work

Fault analysis on electronic systems has always been considered as an important domain as proved by a tremendous collection of references and discussions on this subject during decades following electronic technology progress. However, only a limited number of works have covered RFID systems. Most of these works have focused on the practice of side-channel attacks on RFID tags via several means relying on electromagnetic interferences and optical inductions [1,12–14], while others have practised contact based fault injection techniques [15,1]. Both methods have demonstrated that fault attacks can effectively break cryptographic functions embedded into RFID tags as well as yielding to other errors on tags. These approaches generally evaluate security of manufactured tags against fault attacks but do not discern or identify weak elements on considered tags architectures neither lead to tag architecture improvement. It is important to evaluate tags at design level in order to get and guarantee safe designs before manufacturing steps as it is considered in this paper.

A fault-tolerance evaluation method is presented in [16] where authors have used an FPGA to implement a tag with a fault injection mechanism in addition to an analogue front-end and an external reader. Identical methods are used in [17,18] for fault injection on RFID tags. However, these approaches do not process thoroughly analysis and deliver weak information about injected fault effects. Another work presented in [19] also proposes an FPGA platform (DemoTag) to test tag digital hardware designs and analyse the impact of fault attacks on RFID but without revealing any method for accomplishing it. In our approach, we have used an FPGA to implement the tag under study and we have built a corresponding reader in the same FPGA to get a complete RFID system; thus ready for effective management and monitoring of predefined communication scenarios. Consequently, the behaviour of the RFID system is under control and more trackable.

3. A configurable RFID fault emulation

The fault analysis process consists in checking the response of a circuit in the presence of faults, by comparing the behaviour of both the fault-free and the faulty circuit. Fig. 1 shows the developed FPGA-based fault emulation system. Faults are injected into the experimented tag circuit to check their effects while this tag is in operation. The other tag circuit is kept fault-free to provide comparison means as recommended in testing techniques. The integrated reader in the system has

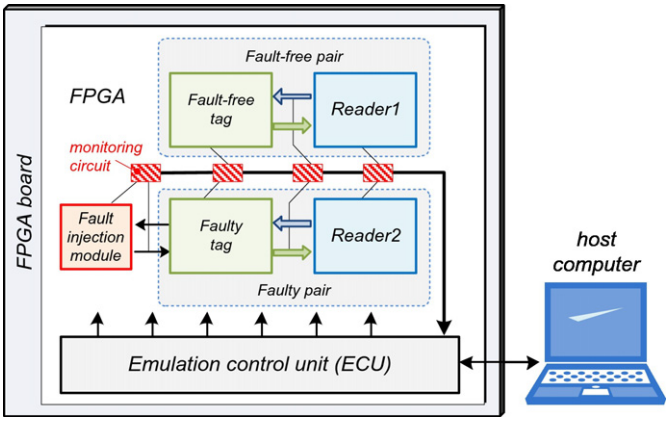

Fig. 1. Developed RFID fault emulation platform.

the role of communicating with the tag. Thus, study real RFID communication is ready with high flexibility of editing scenarios and avoiding generation of stimulus. Reader as a block unit also provides a valuable communication error handling (e.g. errors in the frame encoding and errors to tolerate).

In order to carry out comparison and track injected fault effects, two identical reader circuits are added to form two reader-tag pairs: faulty pair and fault-free pair (Fig. 1). This autonomous communicating tag-reader structure presents appropriate comparison conditions and thus allows full control and monitoring of all RFID operations. The adopted solution excludes the radio frequency (RF) communication through implementing both readers and tags in a single FPGA. That greatly simplifies the emulation process upon keeping valid analysis outcomes since the tag behaviour is willingly maintained unaffected. Notice that getting the same test conditions with external readers is an extremely complicated task since it will require isolation between both RF pairs and sever synchronisation.

As illustrated in Fig. 1, the system also contains additional blocks: *a fault injection module*, *an emulation control unit* and *specific monitoring circuits*. The latter monitor both reader-tag pairs and any occurred communication in order to achieve a comparison. In addition, monitoring circuits are in charge of delivering information about injected faults and their effects on the faulty tag. The resulting global system is configurable, autonomous and thus able to perform complete SEU or SET fault emulation campaign without any external interaction requirement. The emulation process can be speed-up through selecting maximum FPGA frequency. The platform works in interaction with a host computer for loading emulation parameters, acquiring and analysing produced emulation results.

The proposed approach offers the possibility to study any tag architecture compliant with any RFID protocol. For studying a given tag architecture, we should use a corresponding reader circuit with its conforming protocol. In our experimentations on EPC Gen2 protocol, the reader is built around a configurable microcontroller IP-core (more details can be found in Section 3.2). For completing hardware reader part, other required blocks, developed in VHDL, are added to the microcontroller. A simple C program code describing the reader functionalities and supposed scenarios is created, then compiled and incorporated into the readers' program memory. Therefore, generating other protocols than EPC Gen2 is possible using the same reader configuration. This only involves producing the targeted protocol compliant hardware blocks and the suitable C program. Notice that the fault injection module and the emulation control unit are developed in generic format and remain suitable for any protocol.

As far as implementation in FPGA is concerned, we have used Xilinx ISE 14.7 tool to generate our emulation system. The fault injection campaign starts after building the system in the FPGA and loading through the host computers all the emulation parameters (see details in

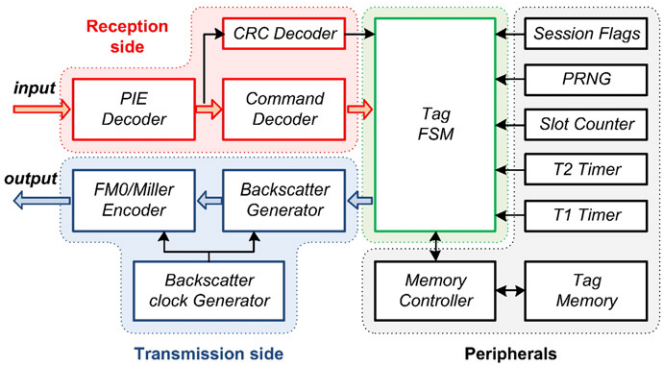

Fig. 2. Experimented RFID tag block diagram.

Section 3.4). A campaign involves several emulation cycles. In every cycle, there is a fault injection, a fault classification and error analysis. The detail of this process is given in Sections 4 and 5. More details on fault emulation platform components are given in the next sections.

3.1. Experimented tag

In this work, a finite state machine (FSM) based tag architecture conforming to EPC Gen2 protocol (approved as ISO 18000-6C) is experimented. Fig. 2 illustrates the architecture with its four main parts organisation: (1) the reception side, (2) the tag state machine (Tag FSM), (3) peripherals, and (4) the transmission side. In addition to the input and the output serial pins (Fig. 2), the tag uses a clock of 2.56 MHz.

Accordingly to EPC Gen2 protocol specifications [8], reader-to-tag communication (R≥ T) carries out through PIE encoding (pulse-interval encoding), while tag-to-reader communication relies on FM0 (bi-phase space) encoding or Miller encoding according to reader recommendations. The reader first interrogates the tag (or tags) by sending supported commands, and then the tag reacts by backscattering responses. Mandatory commands that tags should integrate are: *Select*, *Query*, *QueryAdjust*, *QueryRep*, *ACK*, *NAK*, *Req_RN*, *Read*, *Write*, *Kill* and *Lock*. The used tag integrates all these commands in addition to the optional *Access* command that is devoted to access to the secured state of the tag by sending a password (access password).

All compliant EPC Gen2 protocol tags integrate the seven following states: *Ready*, *Arbitrate*, *Reply*, *Acknowledged*, *Open*, *Secured* and *Killed*. When powered, the not killed tag enters on *Ready* state, a reader can then access to this tag by sending a series of select, inventory and acknowledge commands. Fig. 3 illustrates an example of accessing to a tag and reading the corresponding EPC (electronic product code). For our experimented tag, the whole EPC Gen2 tag functionality including all the tag previously defined states is solely implemented in hardware using VHDL (Very high-speed integrated circuits Hardware Description Language).

The Tag FSM module processes any received command and constitutes the main module which controls the entirely tag functioning. The FSM is organised as a seven *main states* accordingly to the standard [8]. Fig. 4 shows these states and all possible transitions. Since the tag's functionality relies upon the protocol specification and data, the diagram is fully traversed by the protocol. For each FSM main state, several

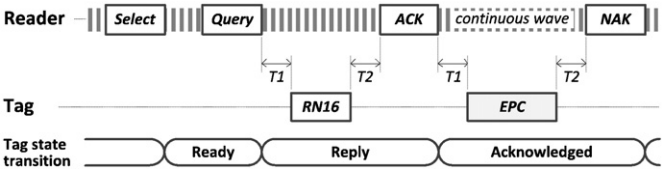

Fig. 3. Reader-tag communication example.

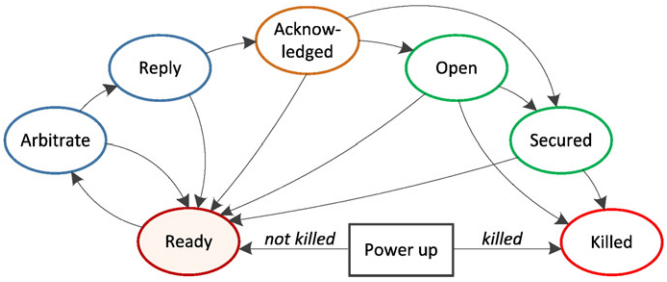

Fig. 4. Experimented tag state diagram.

sub-states are defined to describe any targeted tag functioning. Notice that different sub-states number for each main state and a total of 90 sub-states is derived.

According to the above tag FSM description, it is evident that FSM encoding plays an important role in the robustness and the whole tag reliability since this FSM structure monitors complete tag functionality. For analysing the FSM encoding impacts on the tag, the FSM main states and sub-states are coded with 3 and 5 bits respectively, in concordance to the minimum allowed number of bits.

The implementation of this architecture including the tag memory occupies 1059 slices on Virtex5 (XC5VLX50T), which represents 3.67% of the FPGAs area. This small-devoted area motivates building complete RFID environment integrating several tags.

3.2. RFID reader

The developed platform uses a VHDL RFID circuit reader based on a configurable microcontroller IP-core [20,21]. Multiple reader functions including frames encoding and decoding are then implemented in hardware, thus simplifies processing any mandatory, optional or custom command through only few software instructions. The reader software is developed using C language, it is structured on several functions allowing generation, reception, and processing of frames according to the protocol. For every fault injection analysis, a suitable communication scenario instructs the reader. During this fault emulation process, the reader delivers information about the communication scenario progress and supplies information about any detected error during the communication. This information is collected through the reader output port.

3.3. Fault injection module

Current fault injection techniques are manifold and hence use different resources with a variety of costs. However, fault injection at behavioural-level models is known as a low-cost technique [22,23]. Basing on this technique, a configurable and generic fault injection mechanism is designed and built to process SEU and SET faults emulation. This mechanism relies on a designed configurable fault injection slice (CFIS) which is configurable to inject SEU and SET fault. The use of this slice is illustrated in Fig. 5. Several slices may be inserted to support the targeted vector signals for fault injection (*Data_in*). Every bit of this vector is linked to one CFIS that delivers the corresponding faulty data (*Data_out*); a mask is then used to select corresponding bits for fault injection.

The CFIS comprises two flip-flops (FFs) and some combinational logic as shown in Fig. 6. When enabling fault injection, data bit inversion occurs. The FF1, used as the main circuit to enable fault injection for at least one period of the injection clock (*inj_clock*), is set to '1' when the input signal *inj_enable* is activated. Its function is essential since it isolates the combinational logic delivering *inj_enable* signal. Notice that for each injection enabling, the fault is injected on the next cycle of the clock. In SET mode (*SE_mode* = '0'), FF1 enables the fault injection during one cycle of the clock for each activation of the signal *inj_enable*,

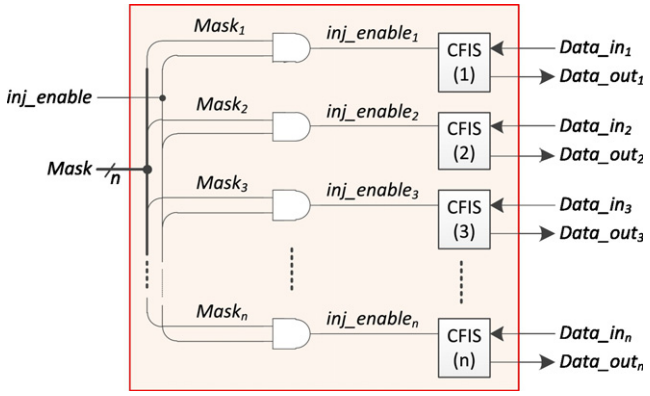

Fig. 5. Fault injection logic structure.

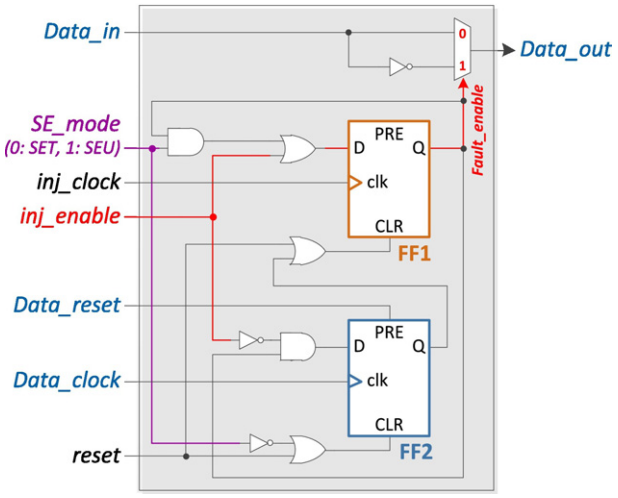

Fig. 6. Composition of the configurable fault injection slice (CFIS).

whereas in SEU mode, for each activation of *inj_enable* signal, FF1 allows fault injection until the next *Data_clock* edge or *Data_reset*, hence provides the emulating SEU effect.

The FF2 concerns only the SEU mode and generates a reset signal for FF1 by disabling the encountered fault injection. This action takes place in the presence of one of the two following cases: (1) when a *Data_clock* edge arrives or (2) when *Data_reset* signal activation occurs. When using CFIS to accomplish SEU fault injection in FFs with clock enable signal, one might add it to FF2 or use it in clock gating. It appears clear for the SET mode case that the FF2 remains disabled. Fig. 7 shows how to use CFIS in the two modes: in the SEU mode, it is necessary to provide *Data_reset* and *Data_clock* with corresponding FF data (Fig. 7a), while in SET mode only fixing *SE_mode* at '0' logic is needed (Fig. 7b).

3.4. Emulation control unit (ECU)

In this developed system, the effective emulation task is carried out under a complete supervision of an emulation control unit (ECU). This latter not only monitors every system components but synchronises the whole emulation process. The role of this unit is therefore multiple, it organises each fault emulation cycle's progress, collects emulation results and finally transfers results to a host computer. The communication with the host computer is achieved through the serial port. A transmission buffer is used for saving and shifting one emulation cycle results. A reception buffer integrated to the unit serves as a storage register for configuration settings. It is loaded from the host computer. The ECU also integrates a specific circuit for performing exhaustive emulation in space and time. This means that it is possible to perform automatically, through one emulation activation, one fault injection at each emulation clock cycle (time exhaustive) and for any bit of the data vector (space exhaustive). The defined exhaustive mode operates separately for time and space, so separate or combined modes are possible. All the emulation parameters including exhaustive mode activation, fault injection mode, emulation duration and others are loadable through the serial port.

4. Fault emulation process

In this designed platform, an emulation cycle processes in four phases (Fig. 8):

The ECU starts by initializing readers, tags and monitoring circuits in the first phase.

1. It enables readers and tags to communicate together for a sufficient period in order to accomplish a desired scenario. Note that during this time and according to stated injection parameters, faults are effectively introduced into the faulty tag. Moreover, the added *monitoring circuits* check and record all-important events.
2. When emulation period expires, the ECU saves yielded results in transmission buffers.
3. Finally, the ECU transfers the data outcome to the host computer.

Notice that any experimentation generally needs several emulation cycles. Therefore, a pipeline is used to shorten the emulation time. As shown in Fig. 8, phases 1 and 2 occur in parallel with phases 3 and 4.

According to injected fault effects on considered scenario evolution, a five-category fault classification is defined as follows:

1. Silent (S): communication scenario achieved with no error detection.
2. Frame alteration (FA): communication scenario achieved in the presence of non-damaging frame alteration.

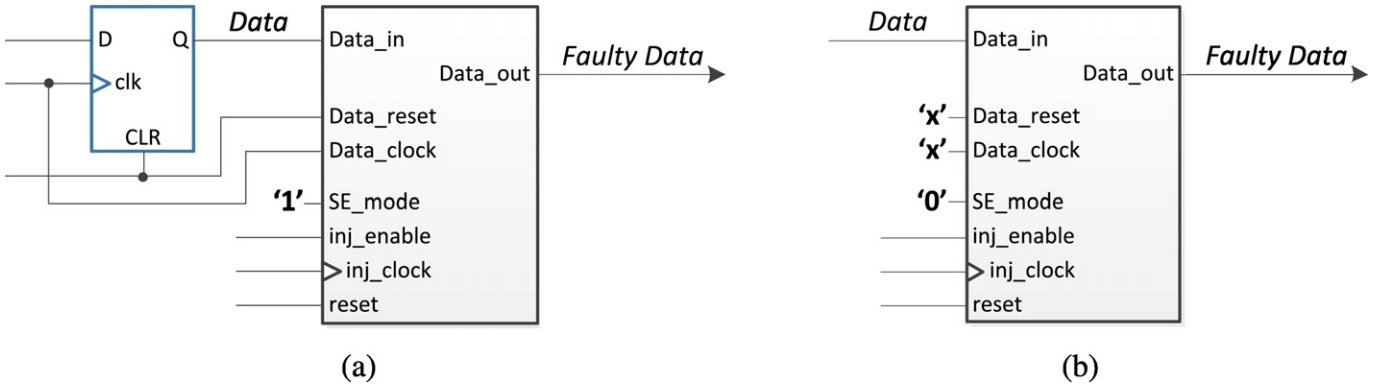

Fig. 7. CFIS use: (a) SEU mode, (b) SET mode.

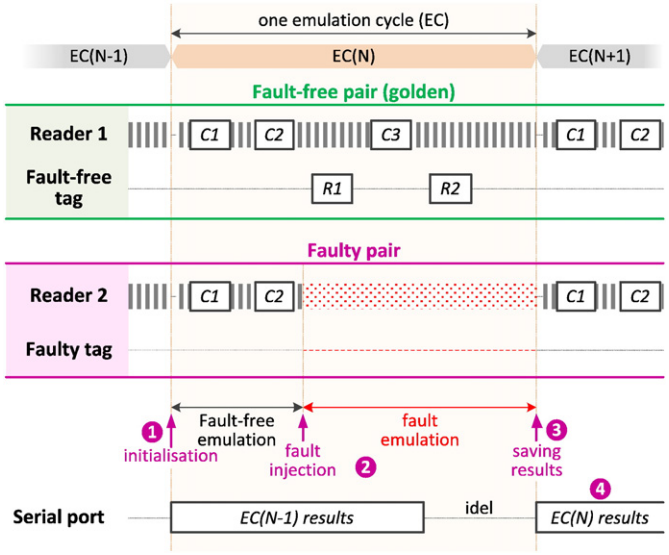

Fig. 8. Fault emulation progress.

3. Latent (L): communication scenario achieved but accompanied by memory content altered which in turn may produce failure on forthcoming cycles.
4. Latent with frame alteration (L&FA): similar to latent but in the presence of frame alteration.
5. Failure (F): no achievement of communication scenario.

It should be pointed out that in typical fault classification, three main categories, *silent* (*masked*), *latent* and *failure*, are generally supported [24–26]. However, in our experimentation on RFID, new unsupported situations concerning frame alteration have led to the introduction of two categories FA and L&FA. This new classification has in fact allowed a better insight of the tag behaviour as demonstrated in the results session.

5. Experimentation

The developed platform supports extensive analysis of SEU and SET impact on RFID tags. In our experiments, we have adopted a typical communication scenario as illustrated in Fig. 9 which gives the command/reply steps detail [8]. For this scenario, an emulation cycle duration takes 8000 tag clock periods where about 7600 periods are sufficient to accomplish the scenario. Notice that the emulation cycle duration should be bigger than the time required for the scenario achievement. This gives enough time to detect eventual unexpected tag activities whose time exceeds scenario time.

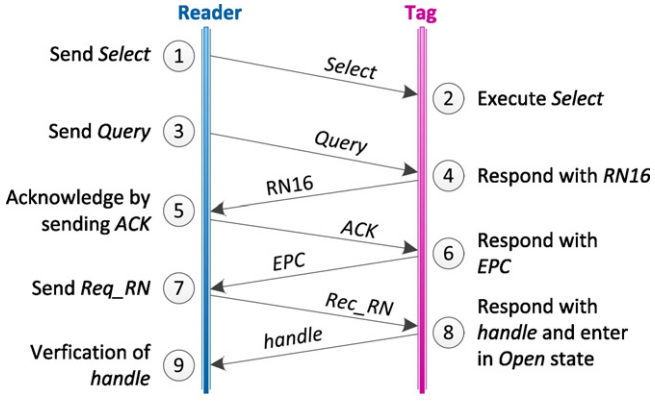

Fig. 9. Experimented communication scenario.

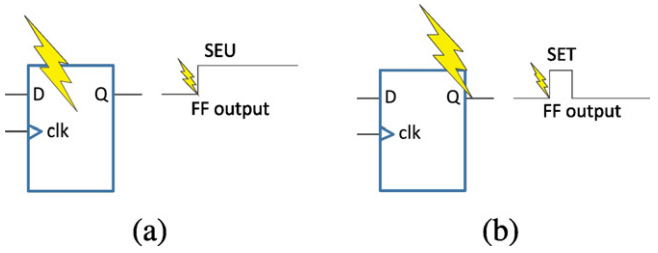

Fig. 10. Illustration of the fault effect on the flip-flop output: (a) SEU effect, (b) SET effect.

In this paper, two experimentations have been carried out for SEU and SET faults independently:

1. In the first experimentation, an exhaustive injection of SEU fault (Fig. 10a) is performed where one SEU is injected for every tag flip-flop at every clock cycle. In total, 314 FFs have been subject to 2,071,772 SEUs in where 6598 SEUs concern each FF. The complete operation consisting of fault introduction and analysis runs automatically.
2. In the second experimentation, we have processed with two SETs fault injection for every FF output (Fig. 10b) and at every clock cycle. So a one-half clock period length SET has been injected exhaustively before active clock edge; and a second SET exhaustive injection occurred after the active edge.

As mentioned above, the two fault injection experimentations targeted all modules that include FFs. Thereby, Tag Memory and Memory Controller modules are not involved since they do not include any FFs. In addition, since injected faults produce the same results when the tag remains in idle cycles, injection results of one cycle is thus a representative of the entire idle cycles.

6. Experimental results and discussion

The previously described experimentations have been carried out on Digilent Genesys board equipped with a Virtex-5 FPGA (XC5VLX50T). The complete system design has consumed 69% of FPGA resources. The fault injection module and the ECU, as shown in Fig. 1, consume 10,982 slices representing then 38% of the used FPGA area. Initially, in the first experimentation, 110 min have been necessary to accomplish the 2,071,772 SEUs emulation using the actual tag functioning clock (2.56 MHz, 3.22 ms/emulation cycle). As the number of emulated SETs in the second experimentation is double, so the experimentation time has consequently doubled (220 min). However, we managed to reduce the emulation time by increasing the emulation clock frequency up to maximum allowed implementation frequency without altering emulation results. The maximum possible frequency for our implementation is 29 MHz, so durations of experimentations using this frequency became 10 min and 20 min for SEU and SET experimentations respectively. The total duration of the fault injection campaign (6,215,316 SEUs and SETs) including results analysis is 30 min. Notice that the yielded experimentation duration is specific to the scenario of Fig. 9 and this duration will increase, as well as the number of injected faults, with longer scenarios. However, the proposed emulation platform and fault classification remain suitable for any scenario.

Fig. 11 illustrates an example of SEU emulation. Shown signals depict the communication of fault-free and faulty pairs during SEU emulation; these signals are obtained via FPGA pins and visualised using a digital oscilloscope. The injected SEU fault in this example has caused a failure (frame corruption) as illustrated in Fig. 11. The numbering of actions in this figure shows an emulation cycle progress in conformity with scenario detailed in Fig. 9.

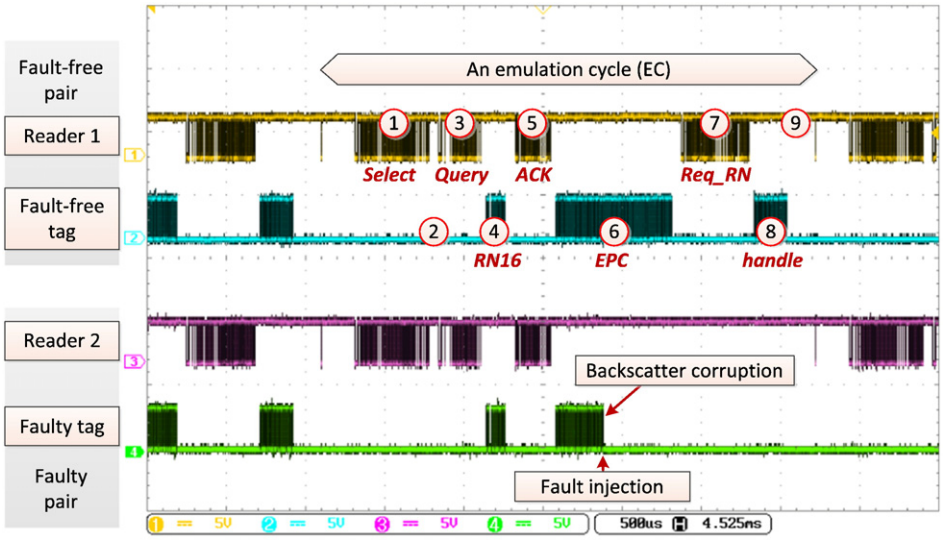

Fig. 11. Fault-free and faulty communication signals during SEU emulation.

6.1. SEUs emulation results

The SEU emulation results are summarised in Table 1. For each experimented tag block, the details about the number of injected SEUs and resulting classification are presented. Analysis of these results shows that the fault effect is linked to each block or side function and in fact, reflects the appropriate integrated functions. For example, higher failures (33%) have been registered in the Tag FSM block since it is in charge of controlling all the other tag parties. In the opposite, detected failures are minimal for the so-called peripherals (under 5%) in concordance with their minor activities during the tag active mode.

On the other hand, silent SEUs are considerable and go up to 71.95% and 75.85% for the reception and the transmission sides respectively. Several signals contained in these two parts are initialised when not used, so the SEU effect disappears following clearance through this initialisation yielding then to a high number of silent SEUs. For SEUs classified under the frame alteration category, their number is limited and mainly concentrated (6.21%) in the transmission side in charge of the frame generation. SEUs which produce latency (latent) and latency with frame alteration (latent & frame alt.) are concentrated on peripherals (31.32% and 26.84% respectively) in agreement with relative changes occurring in PRNG, Slot Counter and Session Flags blocks.

In total, the proportion of SEUs causing system failure is about 16%; the percentage of latent SEUs (L and L&FA) is higher (22.15%) whereas more than 60% of emulated SEUs have a silent effect.

Fig. 12 shows SEU fault classification details where every horizontal line represents a separate tag FF. A full line characterises the total emulated SEUs for any FF (one injection in each clock cycle so 6598 SEUs per line). This depicts an overall SEUs effects distribution and provides a straight identification of critical FFs.

Flip-flops that produce failures are concentrated in the Tag FSM, these FFs are the eight bits of FSM encoding register, one FF generating system reset, the RN16 register (16 bits), FFs storing the *Query* command parameters (*D*, *M* and *TRext*) [8] and finally the four MSB bits of the register that stores TR calibration. Regarding faults producing frame alteration, they appear in a triangle format (with green colour)

Table 1
SEU emulation results.

	Block name	Number of FFs	Number of injected SEUs	Fault classification				
				Failure (F)	Latent & frame alt. (L&FA)	Latent (L)	Frame alt. (FA)	Silent (S)
Reception side	PIE Decoder	42	277,116	38,523	9952	59,252	306	169,083
	Command Decoder	68	448,664	62,402	18,244	31,277	1	336,740
	CRC Decoder	21	138,558	17,692	0	4800	0	116,066
	Total	131	864,338	118,617	28,196	95,329	307	621,889
				13.72%	3.26%	11.03%	0.04%	71.95%
Peripherals	PRNG	16	105,568	0	95,456	10,112	0	0
	Slot Counter	15	98,970	15	0	70,350	0	28,605
	T2 Timer	6	39,588	8	0	0	0	39,580
	T1 Timer	12	79,176	13,388	163	38	2005	63,582
	Session Flags	5	32,990	1906	0	31,084	0	0
	Total	54	356,292	15,317	95,619	111,584	2005	131,767
				4.30%	26.84%	31.32%	0.56%	36.98%
Transmission side	Backs. clock	5	32,990	4599	22,926	66	1542	3857
	Backs. Generator	40	263,920	25,370	216	255	19,575	218,504
	FM0/Miller Encoder	15	98,970	17,615	5	0	3448	77,902
	Total	60	395,880	47,584	23,147	321	24,565	300,263
				12.02%	5.85%	0.08%	6.21%	75.85%
	Tag FSM	69	455,262	150,577	25,770	78,900	2567	197,448
				33.07%	5.66%	17.33%	0.56%	43.37%
Total		314	2,071,772	332,095	172,732	286,134	29,444	1,251,367
				16.03%	8.34%	13.81%	1.42%	60.40%

Fig. 12. SEUs classification for each tag flip-flop.

in the Backscatter Generator. The concerned FFs are those of the 16-bit shift register; this frame alteration number increases linearly from LSB to MSB (most significant bit) and thus the resulted triangle format.

The maximum number of detected silent SEUs is related to the following FFs: T2 Timer FFs, four FFs in T1 Timer, two LSB (least significant bit) FFs of the PW Counter situated in PIE Decoder and six FFs responsible for Miller encoding in FM0/Miller Encoder block where exclusively the FM0 encoding is used in this scenario.

6.2. SETs emulation results

As described above for this experimentation, two SETs are injected in every clock cycle and for every FF. Fig. 13 (a) presents the classification of SETs injected before the active clock edge, while Fig. 13 (b) presents the classification of SETs injected after the active clock edge. The sum of each fault class and its percentage in both cases are given at the top of the figure. For the first case where SETs are injected before the active primary system clock edge, the result is similar to SEUs emulation for the majority of blocks except for Command Decoder, CRC Decoder and Backscatter Generator blocks where the silent faults increased since these blocks use different clocks generated by the PIE Decoder and Backscatter clock Generator, silent faults are then increased by 7%. In the second case where SETs are injected after the active primary system clock edge, only 2% of the injected faults have not silent effect. This portion is related to 13 FFs (Fig. 13b); these FFs are used in general as set/reset signals.

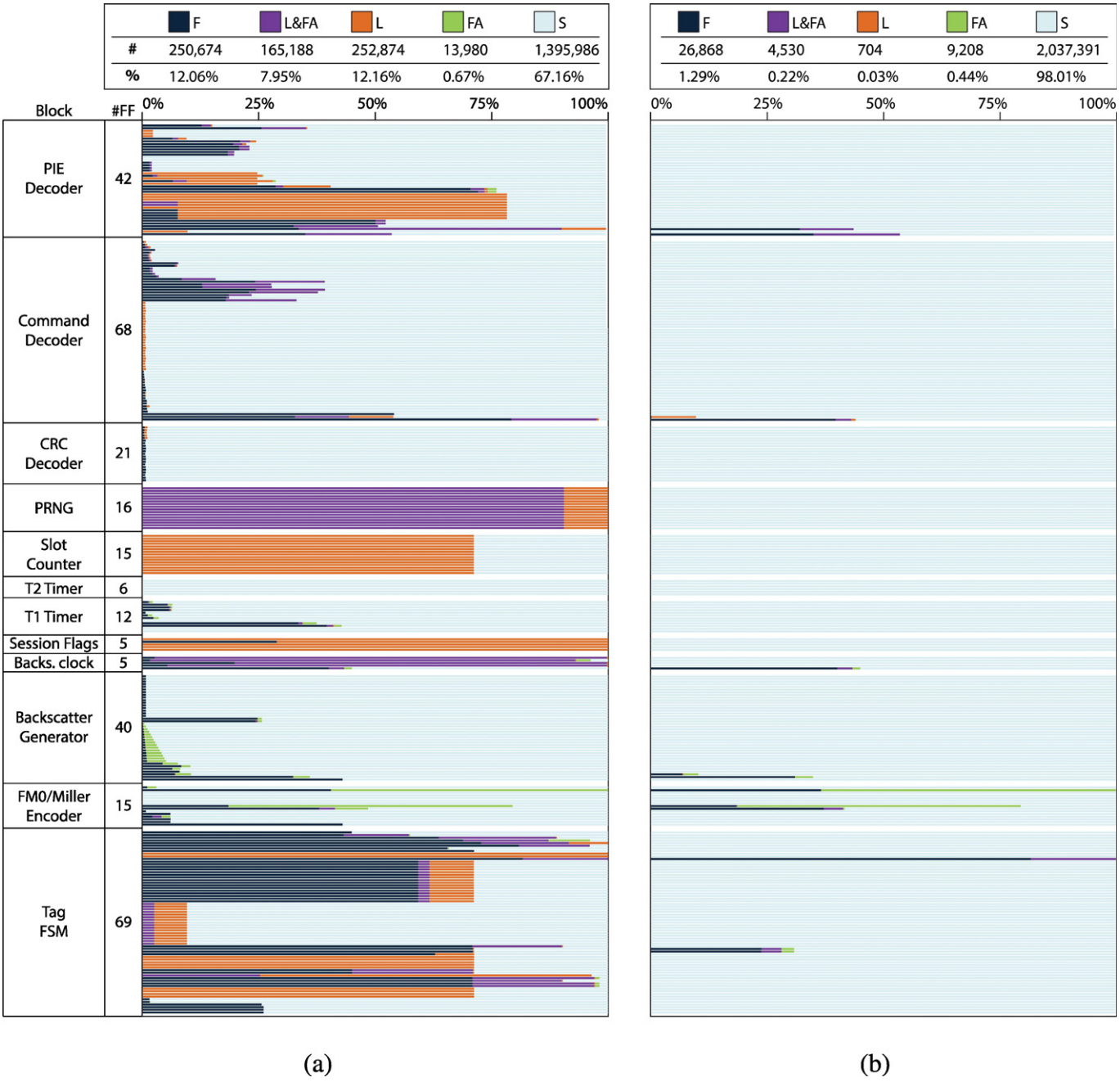

Fig. 13. SETs classification for each tag flip-flop: (a) SETs injected before the active clock edge, (b) SETs injected after the active clock edge.

6.3. Errors analysis

Based on monitoring circuits and readers outcomes, an analysis of occurred errors during the SEUs experimentation has been done. Table 2 summarises the deduced errors, they are divided into two categories, major errors and minor errors, according to their impact on the tag security and reliability. Notice that one fault may produce several errors at the same time.

Minor errors have limited impact on the tag. For example, with 'valid altered frame' error, frames either include non-destructive encoding faults or have delays without any scenario corruption or breaking the communication. The remaining minor errors may be recovered at new interrogations or when initiating a new round.

In addition, the presence of major errors, although in a minority (eight in total), is damaging since the tag security and data contents

Table 2
Summary of produced errors for the SEU experimentation.

Errors		Number	Proportion	Fault category
Major	Enter into the secured state	8	1/258,971	F
	Enter into the killed state	0	0%	F
	Change of tag memory content	0	0%	F
Minor	Valid altered frame	202,176	9.76%	L&FA/FA
	Frame corruption	68,143	3.29%	F
	Invalid frame	40,875	1.97%	F
	Identification failed	157,795	7.62%	F
	Exiting the current round	61,368	2.96%	F
	Change of inventoried parameters	34,926	1.69%	F/L/L&FA
	Change of query parameters	107,085	5.17%	F/L/L&FA

Fig. 14. Presentation of five selected errors from Table 2.

are affected. 'Enter into the Secured state' error is a serious tag security threat since it can lead to secure data delivering or data altering unwillingly. Notice that 'enter into the Killed state' and 'change of tag memory content', considered as major errors, have not been detected through the performed exhaustive injection.

Fig. 14 illustrates the link between five errors selected from Table 2 and tag FFs (filtration of Fig. 12). This representation allows identifying effectively the critical FFs: whenever a FF is responsible of the occurrence of several error types, this FF is considered as critical. Thus, as shown in Fig. 14, FF245 to FF252 (FSM register) and FF255 (general system reset signal) stand for highly vulnerable FFs within the Tag FSM. Among these FFs, FF251 and FF252 are the most critical FFs since they generate the eight SEUs which cause 'enter into the Secured state' error previously depicted (see Fig. 14).

Critical FFs are more easily identified when we carried the superposition of all errors reported in Table 2 for each FF; the result is illustrated in Fig. 15. In this figure, the darker the FF is, the more it is subject to errors. Therefore, critical FFs previously described (eight FFs of the FSM register) and the FF of global reset appear clearly in dark. In addition, Fig. 15 shows other critical FFs, less affected than those of FSM register, situated in PIE Decoder and Command Decoder.

Since major errors occurred, their eradication is mandatory. It is recommended either to redesign sensitive parts if severe modification is targeted, or build new protection means. Increasing FSM encoding length, as well as adding specific monitoring circuits for preventing FSM false transitions may also be considered as suitable alternatives.

7. Conclusion

This work proposes a new FPGA based fault emulation method for RFID tags robustness and security evaluation. The designed configurable platform allows performing SEU and SET impact analysis on tags with flexibility in setting scenarios and related parameters. An EPC Gen2 tag has been experimented under intensive SEU and SET faults with this method not only to provide useful results for improving future RFID designs but also to validate the followed approach to build a versatile FPGA-based platform. This approach is also applicable to other RFID protocols. Fault analysis on experimented tag allowed signalling that 60.40% of SEUs have been tolerated, 16.03% of SEUs caused failures while 23.57% of SEUs produced frame alteration or memory latency.

A classification of produced errors through the emulation system outcomes has permitted evaluating SEU and SET fault impacts. Only

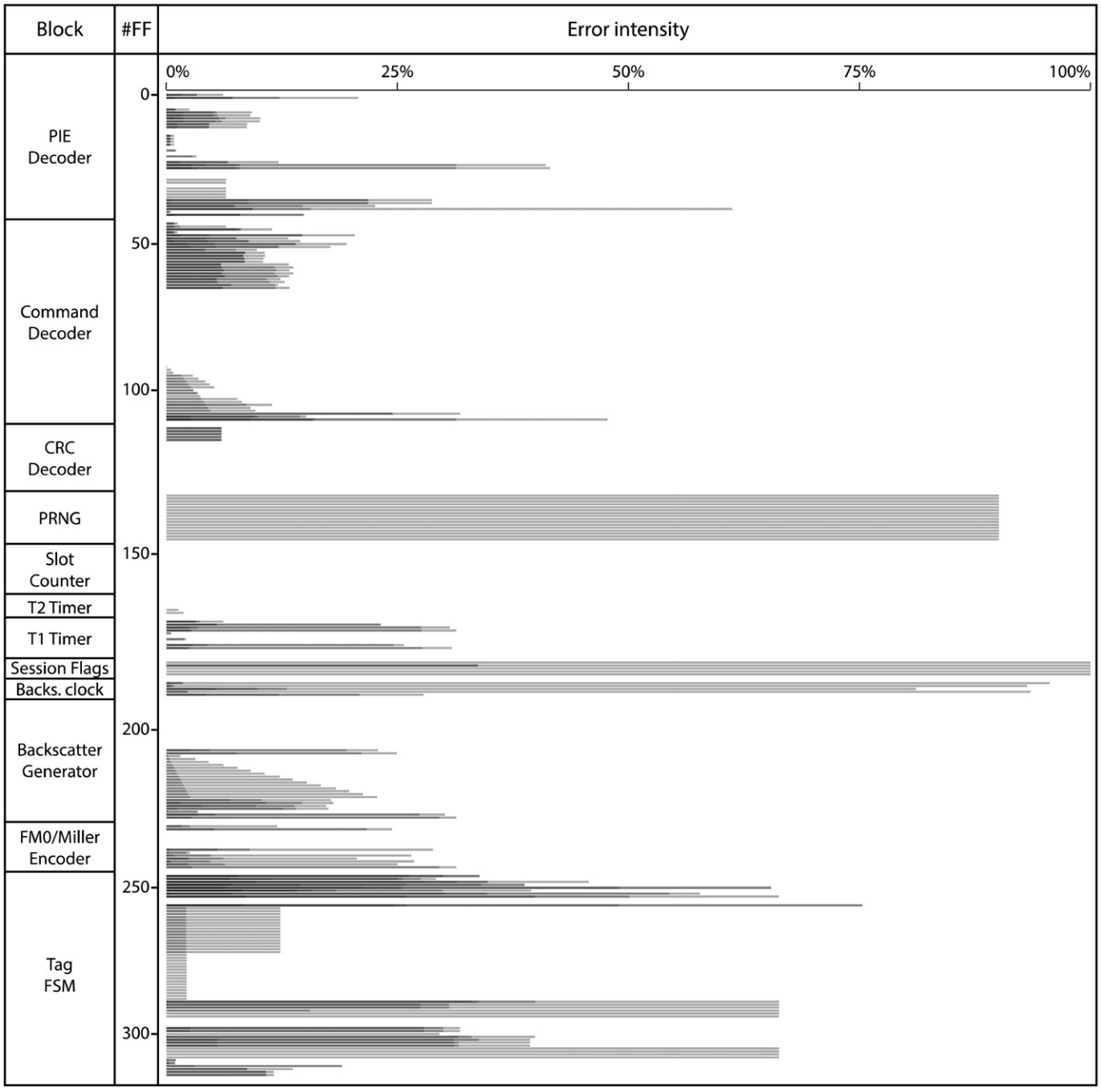

Fig. 15. Error intensity for each tag flip-flop.

eight SEUs in 2,071,772 have caused major errors forcing the experimented tag to enter unwillingly into secured states. Our system has not only identified the concerned major errors FFs but also the most sensitive FFs causing, for instance, a large number of failures. The experimentation has also demonstrated how to derive suitable design actions for low-cost RFID security improvement. Obtained results and produced error classification strategy can be employed to generally enhance RFID diagnosis.

As the actual platform performance permits carrying out multiple event upsets (MEUs), several dedicated campaigns are in process to derive their effects and get more insight about security and robustness. Future work will extend this research to cover multiple other tags' architectures and RFID's standards. Rapid technology changes impose continuous improvement in tag robustness and security, thus strategy through protection means should be more involved and developed.

References

[1] M. Hutter, J.M. Schmidt, T. Plos, RFID and its Vulnerability to Faults, Cryptographic Hardware and Embedded Systems - CHES, Washington, USA, 2008 363–379.
[2] R. Velazco, S. Rezgui, R. Ecoffet, Predicting error rate for microprocessor-based digital architectures through CEU (code emulating upsets) injection, IEEE Trans. Nucl. Sci. 47 (6) (2000) 2405–2411.
[3] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, J. Karlsson, Fault injection into VHDL models: the MEFISTO tool, FTCS-24, Int. Symp. Fault tolerant, Computing (1994) 66–75.
[4] L. Berrojo, F. Corno, L. Entrena, I. González, C. López, M. Sonza, G. Squillero, An Industrial Environment for High-level Fault-tolerant Structures Insertion and Validation, IEEE VLSI Test Symp., Monterey, CA, May 2002 229–236.

[5] L. Antoni, R. Leveugle, B. Feher, Using Run-time Reconfiguration for Fault Injection in HW Prototypes, IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems, 2002 245–253.
[6] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, M. Violante, FPGA-based fault injection techniques for fast evaluation of fault tolerante in VLSI circuits, Forum on Programmable Logic (FPL), Belfast, Northern Ireland, 2001.
[7] C. López-Ongil, M. García-Valderas, M. Portela-García, L. Entrena, Autonomous fault emulation: a new FPGA-based acceleration system for hardness evaluation, IEEE Trans. Nucl. Sci. 54 (2007) 252–261.
[8] EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID, Protocol for Communications at 860 MHz–960 MHz, Version 1.2.0, EPCglobal Inc., 2008http://gs1.org.
[9] A. Ejlali, S.G. Miremadi, Error propagation analysis using FPGA-based SEU-fault injection, Microelectron. Reliab. 48 (2) (2008) 319–328.
[10] M.G. Valderas, L. Entrena, R.F. Cardenal, C.L. Ongil, M.P. García, SET emulation under a quantized delay model, J. Electron. Test. 25 (1) (2009) 107–116.
[11] G. Wirth, F.L. Kastensmidt, I. Ribeiro, Single event transients in logic circuits—load and propagation induced pulse broadening, IEEE Trans. Nucl. Sci. 55 (6) (2008) 2928–2935.
[12] T. Plos, Susceptibility of UHF RFID tags to electromagnetic analysis, Topics in Cryptology–CT-RSA 2008, pp. 288–300.
[13] M. Hutter, S. Mangard, M. Feldhofer, Power and EM Attacks on Passive 13.56 MHz RFID Devices, International Workshop on Cryptographic Hardware and Embedded Systems, 2007 320–333.
[14] A. Barenghi, C. Hocquet, D. Bol, F.X. Standaert, F. Regazzoni, I. Koren, Exploring the feasibility of low cost fault injection attacks on sub-threshold devices through an example of a 65 nm AES implementation, International Workshop on Radio Frequency Identification: Security and Privacy Issues 2011, pp. 48–60.
[15] M. Hutter, J.M. Schmidt, T. Plos, Contact-based fault injections and power analysis on RFID tags, European Conference on Circuit Theory and Design, ECCTD 2009, pp. 409–412.
[16] O. Abdelmalek, D. Hely, V. Beroulle, Emulation based fault injection on UHF RFID transponder, 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems 2014, pp. 254–257.
[17] O. Abdelmalek, D. Hely, V. Beroulle, I. Mezzah, An UHF RFID emulation platform with fault injection and real time monitoring capabilities, 8th International Design and Test Symposium (IDT), IEEE 2013, pp. 1–2.
[18] I. Mezzah, O. Kermia, H. Chemali, O. Abdelmalek, V. Beroulle, D. Hely, Assertion Based On-line Fault Detection Applied on UHF RFID Tag, 8th International Design and Test Symposium (IDT), IEEE, 2013 1–5.
[19] T. Plos, M. Aigner, T. Baier, M. Feldhofer, M. Hutter, T. Korak, E. Wenger, Semi-passive RFID development platform for implementing and attacking security tags, Int. J. RFID Secur. Cryptogr. 1 (2012) 16–24.
[20] I. Mezzah, H. Chemali, S. Mezzah, O. Kermia, O. Abdelmalek, MCIP: high configurable 8-bit microcontroller IP-core, Science and Information Conference (SAI), IEEE 2015, pp. 1387–1390.
[21] MCIP_open, Hyperlink: http://opencores.org/project, mcip_open.
[22] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, J. Karlsson, Fault injection into VHDL models: the MEFISTO tool, 24th International Symposium on Fault-Tolerant Computing, IEEE 1994, pp. 66–75.
[23] P.K. Lala, Transient and Permanent Fault Injection in VHDL Description of Digital Circuits, Circuits and Systems, 3, 2012 192–199.
[24] P. Civera, L. Macchiarulo, M. Rebaudengo, M.S. Reorda, M. Violante, Exploiting FPGA-based techniques for fault injection campaigns on VLSI circuits, IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems 2001, pp. 250–258.
[25] M. García Valderas, M. Portela García, C. López, L. Entrena, S.E.U. Extensive, Impact analysis of a PIC microprocessor for selective hardening, IEEE Trans. Nucl. Sci. 57 (2010) 1986–1991.
[26] M. Ebrahimi, A. Mohammadi, A. Ejlali, S.G. Miremadi, A fast, flexible, and easy-to-develop fpga-based fault injection technique, Microelectron. Reliab. 54 (5) (2014) 1000–1008.