Fault Tolerance Evaluation of RFID Tags

Omar Abdelmalek, David Hély, and Vincent Beroulle
LCIS
Grenoble Institute of Technology
Valence, France
Email: firstname.lastname@lcis.grenoble-inp.fr

Abstract—In order to increase the robustness of a RFID digital circuit against SEUs, fault injection is commonly used to locate weak areas. In circuit-emulation it is a very powerful tool to locate these areas by executing huge fault injection campaigns. In this work, fault injection has been extensively applied to the digital baseband of an UHF RFID tag during the communication with a RFID reader. A large number of fault campaigns have been performed in order to identify the most sensitive parts in the digital baseband. Following this analysis, a first low cost countermeasure is introduced and validated.

Keywords—Robustness, SEU, RFID, hardware fault injection

I. INTRODUCTION

RFID systems are increasingly used for critical applications in harsh environment. Such systems may be used for traceability purpose in many critical application such nuclear plants, aeronautics or other industrial applications. In this context, defaults in traceability may have dramatic consequences, for instance contaminated tools may bypass sanitary checking. A tag when defective leads to RFID detection and identification failures and thus derogates the availability, the reliability and the safety of the whole RFID systems.

RFID is a wireless technology that allows for automated remote identification of objects. The major components of an RFID system are tags or transponders that are affixed to objects of interest and readers or interrogators that communicate remotely with the tags to enable in this work, we propose to study the robustness of RFID UHF tags[1] against single event upset (SEU) in order to increase their robustness at the lowest cost. Since these tags are used in harsh environments, they are particularly exposed to such SEU effects [2]. Studying SEU sensitivity of such circuits raises several issues. First, RFID systems are complex and heterogeneous systems. It is thus necessary to study the behavior of a tag in presence of faults. But it is also important to analyze the effect of a faulty tag on the whole RFID systems. For instance a faulty tag may disturb the system in such way that the other tags can no more be accessed by the reader. Also, although RFID EPC global Class 1 Gen 2 (EPC C1 G2) protocol is a standardized protocol for UHF RFID systems, many different ways to perform inventories for traceability purpose may be developed by reader providers. In fact, all these systems aim at inventory at a given time the tags in front of the reader. Nevertheless the way the inventory is done (although it uses standardized commands) may differ from a system to another. Thus, in order to design a so-called robust RFID tag, it is necessary to evaluate its weakness and the proposed countermeasures within a third party system (i.e. commercial reader and tags).

We propose then in this work to perform and analyze a fault injections campaign on EPC C1 G2 RFID tags within a real RFID system. The fault injection campaign is carried out using the RFID EPC C1 G2 emulator proposed in [6]. The emulator with fault injection capabilities is merged within a system compounded of a third party reader with its proprietary inventory software and third party RFID tags. Effects of injected faults are first measured on the single emulated tag, and then the faults effect is measured on the whole system characteristics. The paper is divided as follows. Section 2 presents the EPC C1 G2 standard and the main characteristics of a tags inventory. Then, section 3 discusses the faults injection campaign set-up detailing how the faults parameters and the way the fault effects are measured. Section 4 details the results of fault injection campaigns in order to identify the main critical parts of an RFID tag. In section 5 a simple countermeasure is proposed taking into account previous campaigns results and the effects of this countermeasure are experimentally measured. Finally conclusions and future work are discussed in section 6.

II. EPC CLASS-1 GEN2 PROTOCOL AND ARCHITECTURE

A. EPC Class-1 GEN2 Protocol

The EPC global Class-1 Generation-2 standard [1] is a standard protocol involved into passive UHF RFID systems (800 -960 MHz). In these systems, the readers transmit continuous wave to remotely power on tags as illustrated in Fig. 1. This means that a tag does not have any embedded energy source. Communication ranges are from 2 to 10 meters. The tag answers to reader successive commands to finally transmit its identification data which is used to identify an object.
Inventory operations are based on the slotted Aloha collision resolution. To avoid collision during communication with the reader, each tag randomly chooses a random time slot. If only one tag chooses the 0 time slot, then this tag reply immediately and all the remaining tags say nothing. The reader, after either receiving a reply or no response from a tag, issues a QueryRep command, causing all the tags to decrement their time slot number (or counter) by 1. Then, any tags reaching the time slot 0 responds. If the number of slots is chosen properly, one and only one tag will respond to most of the QueryRep commands. But if two or more tags respond during the same slot then there is a collision. In this case, a new random slot must be chosen by these tags to communicate. The command Query request allows the reader to start a new inventory round in which all the tags randomly choose a new time slot. If the number of slots is not chosen properly, the QueryAdj command permits to increase or to decrease the total number of slots. When the tag number is very high, the inventory of all the tags can necessitate multiple successive QueryAdj commands. QueryAdj command is used to adjust the Q value parameter. This parameter is detailed in the following section.

EPC C1 G2 operates as shown in figure 3 in which each step is detailed:

1. Reader sends a Query request message to the tags.
2. Each tag which is receiving the Query request generates a 16-bit random number value (RN16) using an internal Pseudo-Random Number Generator. After that, each tag uses this random value (RN16) to select a slot counter. A tag with a slot counter equal to 0 sends its RN16.
3. As response, the reader sends back the ACK command with the random value RN16 in a reserved field.
4. The tag which receives ACK compares the random value in the ACK command with its RN16. If the values are the same, the tag sends several information to the reader: the PC (Protocol-Control containing Physical layer information), the EPC (Electronic Product Code identifying the tag), and a CRC (Cyclic Redundancy Code for error detection). Once a tag has been detected and has sent its EPC, it stops participating until a new inventory is engaged.

B. EPC C1 G2 tag digital baseband architecture

Fig. 3 shows the proposed architecture of our digital baseband circuit. This architecture intensively described in [7] is compliant with the EPC C1 G2 UHF RFID protocol.

The system consists of 12 modules: Pulse Interval Encoding (PIE) decoder, command decoder, tag FSM, CRC decoder, session and flag managers, Pseudo Random Generator (PRNG), slot counter, memory controller, tag memory, backscattering generator, backscattering clock generator (BLF) and FM0 Miller encoder. The PIE decoder decodes the signal coming from the RF demodulator (described in the following) and extracts the data rate and all the timings communication parameters included in the received signal generated by the reader. The command decoder contains an array with all the supported commands. After receiving the signal from the PIE decoder the command decoder compares it with this array in order to detect all the EPC C1 G2 standard commands. The FSM executes the EPC C1 G2 protocol controlling the whole system components (for example controlling the data exchange with the memory). CRC decoder is used by the FSM for CRC5 and CRC16 computing to verify the reader commands integrity and to respond to the reader with CRC protection against transmission errors. The session and flag module controls all the flags affected by the reader Select command during tag selection phase [1]. PRNG module is used by the FSM for generating random numbers required by the protocol. In fact, based on the EPC C1 G2 protocol, two random numbers are needed during the communication process.
One is a random number used to choose the timing slot used for the tag response and the other is a 16-bit random number (RN16) used for communication with reader (as illustrated in Fig.2).

Since the main purpose is to identify the most sensitive data processed by the tag, it is first necessary to describe these data considering the EPC standard. In the following we give definitions of EPC C1 Gen2 standard parameters by explaining their role.

- TRcal and DR: Tag-to-Interrogator calibration symbol and Divide Ratio, these parameters are used to specify the Tag’s response frequency (called backscattering frequency).

- RTcal: used to calculate the threshold time to differentiating an Reader to tag data-0 symbol from a data-1 symbol, also used to compute different Link timing parameters. This parameter is used by the tag to meet all the timing requirements during the communication with the reader.

- Command: this register store the command identifier sent by the reader.

- TRext and M: using these parameters the tag selects the encoding format to answer to the interrogator commands.

- CRC5 and CRC16: cyclic-redundancy checkers that checks the validity of reader to Tag commands.

- Slot counter and Q: Parameters that the reader uses to regulate the probability of a tag to respond. Reader commands all the tags at the beginning of each inventory round to load a Q-bit random (or pseudo-random) number into their slot counter. The Q parameter is adjusted thanks to the Query Adj command.

- Query session, and target: they are flags indicating whether a tag may respond to an interrogator inventory.

All these parameters are controlled by the reader, except the slot counter which is generated by the tag based on the received parameters or commands.

III. RFIM : PLATFORM FOR FAULT INJECTION IN RFID TAG

A. Fault Injection Platform

In order to study the effects of SEU on RFID systems, we use the emulation platform RFIM described in [6] and [7]. The emulation board contains a FPGA, and a communication interface between the board and the host computer. The FPGA acts as a central component of the fault emulation system. Besides the RFID digital baseband under Test, the FPGA also contains a fault injection module and a controller managing the emulation process. Emulation based fault injection is particularly adapted to RFID systems. Thanks to the emulation platform we can perform intensive fault injection campaigns on the digital baseband. Then the fault effects can be evaluated considering the whole RFID system components interactions. For this work, we are using a third party reader with its own inventory protocol. For a fixed period, the reader inventories the tags within its field. During this fixed period, the reader performs successive inventories adapting the Q parameter depending on the number of tags (or detected collision). At the end of this period, the reader indicates how many times each tag has been inventoried. Thanks to the emulation, it is then possible to measure the fault effect on:

- The faulty tag (i.e. how the number of times the tag is inventoried varies with the faults)
- The other tags within the system (i.e. how the number of times the other tags are inventoried varies with the faults)

B. Injected Faults

The main EPC C1 G2 parameters have been detailed in the previous section. As a first approach we have decided to inject SEU [2] only in the registers storing these parameters in the digital baseband.

The faults have then been injected considering each parameter separately. A fault injection campaign has then been carried out targeting each of the register. The faults are injected using the mechanisms described in [6]. In this paper, we only focus on SEU (i.e. single bit flip). For each fault campaign, successive SEUs will be then injected within a single parameter during the whole time of the tag interrogation time (which includes many inventories). The main parameters of the faults injector are:

- The injection rate
- The fault locality within the targeted register

The fault locality within a register is random while the fault injection rate has been fixed.

We have chosen a high injection rate, to have a significant effect that allows us to easily locate the sensitive registers to faults injection. We select a rate of 70% by flipping 7 bits in different positions randomly determined every 10 clock ticks as it is explained in [6]. It is to be mentioned that using a lower rate, the same registers will also be highlighted as the most sensitive ones. Indeed, using a low injection rate, the time between two faults injection is long, which allows the tag to rewrite all the registers, and calculates new parameters. Then with low injection rates the tags do not respond correctly only in some rare cases in which the injection occurs exactly when the tags receive or calculate new parameters. Therefore injection campaigns using low injection rates only decrease slightly the number of times tags are identified. As we have seen, the slotted Aloha collision resolution is by nature
random, then the number of times each tag is identified during a fixed period varies (even without fault injection). Thus a too low impact of the fault injection does not allow us to determine the sensitive registers.

C. Fault Effect Evaluation

A program measures the number of times the tags are detected during the inventory time. The inventory time is fixed to 200s and the time between the end and the beginning of the next inventory round is fixed to 2 us. Therefore the number of times a tag is detected depends on the duration of the response of each tag and also depends of the number of tags in the inventory field. Each injection campaigns are made several times and the final result is computed by averaging each campaign results (It is to be mentioned, that the results are stable from a campaign to another). We define here two ways to measure the fault effects:

- The number of successful tag identification: this indicates the number of times each tag has been inventoried during the 200 seconds spent in front of the reader
- The fault impact on the identification: this value is expressed in percentage, it indicates the evolution of the number of times the tag has been identified with the presence of faults against the result obtained with a non faulty tag.

IV. FAULT INJECTION CAMPAIGN RESULTS

A. Standalone Tag Evaluation

A first fault injection campaign has been carried out on a single tag (the emulator) in front of the reader. Fig. 4 hereafter gives the influence of the fault injection on the number of times the tag has been successfully identified. Light gray gives the value in case no faults are injected; dark gray gives the resulting number in case faults have been injected within the parameters given in horizontal axis.

![Figure 4: Successful Tag Identifications](source)

At a first glance, we can see that all parameters are not equally sensitive. While some faulty parameters make the number of successful tag identification decrease from 4500 to less than 500, other ones have a very limited influence on the tag response. This can be explained by the role played by each parameter during an inventory round, and the refreshment rate of the parameter value during the same round. For instance, when we inject faults in the register RTcal we see a significant drop in the number of tag reading from 4500 to 244. The RTcal parameter sensitivity is explained by the fact that the RTcal data is used to compute pivot duration value by dividing the RTcal duration by 2, this value is a decision threshold differentiating a data-0 symbol from a data-1 symbol, an error in this register causes an error in frame and the tag does not properly decode the command sent by the reader, therefore do not answer.

This is also the cases for other parameters which are directly used to compute the protocol data. This also explains the drop in number of tag reading from 4500 to 542 in the case of DR parameter fault injection. In the case of the TText parameter, the tag completely changes its response preamble, which is different to what the reader requests. Thus, the reader does not detect the response and does not send the Acknowledge command. Finally thanks to our emulation platform the critical registers can easily be identified. As a first conclusion, it can be noticed that the most critical registers in the design are those related with computing the backscattering frequency i.e. RTcal, DR, TText and M, and the register that manages tag population management as query session.

Concerning the less sensitive registers, most of them are not sensitive to fault injection because they are rewritten at each inventory round, and are used for only one clock cycle during a round. Obviously the most used parameters during a round are the most sensitive ones. For this reason the fault injection have an impact only on often used registers. Tag evaluation within a multiple tags system

A second fault injection campaign has been carried out measuring the fault effects on other tags. For these experiments the faulty tag has been put in front of the reader with other commercial tags. This experiment allows us measuring how the faults injection in our tag emulator can impact the performance of a complete RFID system based on several tags. In fact, although the faults are only injected in our tag emulator, other tags not directly concerned by the fault injection can be disturbed by the faulty tag present in the same reader field. For example, we know that if the faulty tag wrong behavior involves the tag always answers to the reader queries then no other tags can be detected. So, we want to verify if four commercial tags used in presence of our faulty tag emulator have the same read rate. Fig. 5 hereafter gives the influence of a fault on the number of times the tag has been inventoried for each possible faulty parameter and for each tag in front of the reader. When the percentage is above 100, it indicates that the tag has been more inventoried, in the other case; it indicates the faults induce a decrease of the number of times the tag has been inventoried. We can see that the results corresponding to the faulty tag are similar to the ones obtained in case the tag is in standalone. Concerning the fault free tags, we can observe two different behaviors. For tag 2 and tag 3, the faulty tag induces an increase of the read rate, while for tag
1 and 4, most of the time it induces a decrease. Also it is to be noted that the increase or decrease never exceed 15 per cent. This can be explained by several points. First the faults which are injected in the faulty tags are not fault which makes the faulty tag sending data in loop to the reader. Such faults would make impossible the communication with other tags and thus would induce a global decrease. Secondly, the faults which are injected make the faulty tags answering with a bad timing to the reader or with a wrong data. Thus the faulty tag not processed by the reader involves more slots to communicate with reader for the other tags during the fixed time spent in front of the reader. Some tags will then use these slots efficiently to communicate with the reader, while other tags will not use it (this is done using a random number generated on chip, as explained in section 2).

These results highly depend on the software embedded in the reader. In fact, according to the algorithm used to fix collisions, these results may vary. Also a future work will focus on performing fault injection for different commercial reader with third party algorithm.

V. ERROR DETECTION AND CORRECTION

Thanks to the previous experimental results, the most sensitive data have been identified: these data are the RTCal, DR, TRext, and Q_session. We propose in a first approach to use hardware redundancy to decrease the fault effects. A triple modular redundancy has been applied. Since TMR is area consuming it is used on the most sensitive identified registers DR, TRext, Query_session. Moreover as such registers are very small this makes the cost acceptable. As shown in Figure 6, the TMR [4] technique consists on the triplication of the target component to protect.

The three resulting outputs from triplication are connected to a voter block that compares the three received data and elect that of majority. If one of the three components fails or suffers a direct SEU then the value contained in the two other registers will be used. TMR technique implies an area increase of the redundant part of more than 200% due to the component triplication. It also needs a voter that is implemented just with some OR and AND gates for each bit of the triplicated component. Since the RFID digital baseband is powered wirelessly and has a limited resource, the TMR was chosen to
protect only the most sensitive registers. Fig. 7 hereafter gives the read rate evolution in case of errors on the sensitive tag registers with TMR. As expected, the rate evolution is almost constant, since the errors are masked.

We note that the use of TMR improves the read rate in the presence of faults. The tag responds with the same number of times in the presence of faults that proves that TMR efficiently protects the sensitive registers. The added surface for embedding this TMR corresponds to 30 flip-flops (FF): tripling of the RTcal adds 18 FF plus 3 FF for the DR register, 3 FF for the TRev register and 6 FF for the Query_session register. We have implemented the tag RFID on Xilinx Spartan-3 device which is low cost FPGA. Table I shows the results reported by the Xilinx ISE tool after the place and route.

| TABLE I. DEVICE UTILIZATION TABLE FOR THE TAG WITH TMR AND TAG WITHOUT TMR |
|----------------------------------|--------------------------|
| # Slice Registers               | TAG Without TMR: 382     |
| # Slice LUTs                    | TAG using TMR: 380       |
| # LUT-FF pairs                  | TAG Without TMR: 782     |
|                                  | TAG using TMR: 810       |

TMR although expensive is in this case an acceptable method since thanks to the fault injection campaigns the most sensitive elements have been identified in a real RFID context, limiting the TMR uses to only a few bits.

VI. CONCLUSION AND FUTURE WORK

We have presented in this work practical experiments concerning SEU evaluation of RFID tags digital baseband in a commercial RFID systems. The emulator is very useful to consider the fault effects even if the reader implementation is unknown. The experiments carried out have led to the identification of the most sensitive data regarding a usual RFID application (i.e. tags inventory). The dedicated RFID emulation platform has been proven to be an efficient tool in order to evaluate fault effect in a real heterogeneous RFID system. Thanks to the experimental results, we have been able to define at the lowest cost countermeasure against potential transient errors in the RFID digital baseband, avoiding the heavy implementation of error detection and correcting code. In this particular case, where a few data are sensitive for the tag identification, we show that a simple TMR on sensitive data can be enough to protect the tag digital baseband. Given the size of the sensitive register of digital baseband, the TMR area and power cost is acceptable in comparison with other technique such ECC for instance. Future work will focus on larger fault injection campaigns on the whole baseband including all the flip-flops, the injected faults will not be restricted to SEU, MEU will also be injected. Also, since the read rate highly depends on the reader software, the campaigns will be carried out using several commercial readers with third party algorithm and considering also more tags.

REFERENCES

[1] EPCglobal, EPC radio frequency identity protocols classe-1 generation-2UHF RFID, protocol for communications at 860 MHz 960 MHz, version1.2.0, 2008.