Design and prototyping of robust architectures for UHF RFID Tags

Omar Abdelmalek

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Nanoélectronique et Nanotechnologies.**

Arrêté ministériel : 7 août 2006

Présentée par

Omar ABDELMALEK

Thèse dirigée par **Vincent Beroulle** et
codirigée par **David HELY**

préparée au sein du **Laboratoire de Conception et d'Intégration des Systèmes (LCIS)**
dans **l'École Doctorale EEATS**

Conception et prototypage d'architectures robustes de tags RFID UHF

Thèse soutenue publiquement le **20 octobre 2016,**
devant le jury composé de :

M, Bruno ROUZEYRE
Professeur des Universités, Université Montpellier II, Rapporteur
M, Romain, OLIVIER
Professeur des Universités, Université de Cergy-Pontoise, Rapporteur
M, Nacer, ABOUCHI
Professeur des Universités, CPE LYON, Président
M, Vincent, Beroulle
Maître de Conférences, Grenoble INP, Directeur
M, David, HELY
Maître de Conférences, Grenoble INP, Co-encadrant

English Title:

DESIGN AND EMULATION OF ROBUST ARCHITECTURES FOR UHF RFID TAGS

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisors Prof. Vincent Beroulle and David Hély for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Bruno ROUZEYRE, Prof. Romain OLIVIER, and Prof. Nacer ABOUCHI, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I thank Christophe Medina (with whom I shared the joys and galleys of the prototype test at RFT LAB platform).

Last but not the least, I would like to thank my family: my wife my daughter INAYA, to my parents and to my brothers and sister for supporting me spiritually throughout writing this thesis and my life in general.

Finally, I would like to thank my great mother whom I lost during my thesis, and these prayers for me.

Résumé:
Les systèmes RFID sont de plus en plus utilisés dans des applications critiques fonctionnant dans des environnements perturbés (ferroviaire, aéronautique, chaînes de production ou agroalimentaire) ou dans des applications où la sécurité est essentielle (identification, lutte contre la contrefaçon). Pourtant, ces systèmes faibles coûts sont peu robustes par nature. Pour les applications critiques, les défaillances des tags RFID peuvent avoir des conséquences catastrophiques ou créer des failles de sécurité importantes. Ces défaillances peuvent avoir des origines nombreuses : par exemple, des origines matérielles dues au vieillissement naturel des circuits intégrés ou à des attaques.

L'objectif de la thèse est d'accroitre la robustesse des tags UHF passifs en proposant et validant de nouvelles architectures numériques de puces RFID robustes à la fois aux défaillances et aux attaques matérielles. Les approches de durcissement des circuits intégrés étudient généralement leur robustesse par simulation et ce de manière indépendante à la validation de leur conception. La méthode la plus courante afin de valider la robustesse d'un circuit repose sur l'injection de fautes par simulation. Pour les tags RFID, ce type d'approche par simulation est problématique car les performances des puces dépendent de nombreux paramètres difficilement modélisables globalement. En effet, le fonctionnement d'un tag dépend de son environnement électromagnétique, du nombre de tags présents dans le système, des protocoles mis en œuvre, l'objectif de cette thèse est développer une méthodologie basée sur le prototypage permettant d'éviter des simulations complexes.

Une plateforme RFIM d'émulation de tag RFID à base FPGA qui conforme à la norme EPC C1 Gen2 a été développée, l'émulateur du tag RFID a été validé fonctionnellement dans un environnement réel. La plateforme RFIM utilise la technique d'instrumentation permettant l'injection de fautes dans les circuits numériques sur FPGA. Grâce à des campagnes d'injection de faute émulées La plateforme RFIM permet d'analyser l'effet sur l'ensemble du système des fautes injectées dans le tag, et valider de nouvelles architectures numériques de tags RFID robustes. La plateforme RFIM a été a été notamment utilisée pour démontrer les effets de nouvelles attaques contre les systèmes RFID reposant sur l'insertion de tags fautifs ou malveillants qui contient un cheval de Troie matériel dans les systèmes. Finalement, cette plateforme d'émulation mène au développement d'une contre-mesure contre les effets des fautes. Cette contre-mesure a été mise en œuvre et évaluée dans un environnement réel avec un lecteur RFID et plusieurs RFID tags.

Mots-clés :
RFID, FPGA, Injection de fautes, Sécurité matérielle, Prototypage, Robustesse matérielle

Abstract
RFID tags are more and more used for critical applications within harsh environments (aeronautics, railways) or for secure applications such as identification, countermeasure against counterfeiting. However, such low cost systems, initially designed for non-critical applications with a high volume, are not robust by themselves. For critical applications, a malfunction of RFID chip may have serious consequences or induce a severe security breach for hackers. Dysfunctions can have many origins: for instance, hardware issues can be due to aging effects or can also be due to hackers attack such as optical or electromagnetic fault injection. It is thus a common practice for critical applications to increase the robustness of RFID system. The main purpose of this PhD Thesis is to increase UHF tags robustness by proposing new digital architectures of RFID chips which would be resilient against both hardware attacks and natural defects.
Usual design techniques for robustness IC improvement consist in evaluating the design robustness by simulation and to do this independently of the design validation. The main technique for robustness evaluation is the simulation based faults injection. Within the RFID context such an approach only based on simulation has several drawbacks. In fact, simulations often are inaccurate because the system behavior relies on several parameters such as the global electromagnetic environment, the number of tags present in the reader field, the RFID protocol parameters.
The purposes of this PhD are to develop a design method dedicated to RFID system based on hardware prototyping in order to avoid time consuming simulations and then to evaluate the design within a real environment.
The hardware prototyping based on FPGA allows the design to be validated in a real environment. Moreover, using instrumentation techniques for fault injection within FPGA, it will be then possible to analyze the effects of faulty tags on the global system in terms of safety and security and then to propose countermeasures.
In this thesis an FPGA based emulation platform called RFIM has been developed. This platform is compliant to EPC C1 Gen2 RFID standard. The RFID tag emulator has been validated functionally in a real environment. The RFIM platform uses the instrumentation technique for injecting faults in the digital tag circuit. Through fault injection campaigns RFIM platform can analyze the effect on the entire system of the faults injected into the tag, and ten validate new robust digital architectures.
The RFIM platform has been used to demonstrate the effects of further attacks against RFID systems based on the insertion of faulty or malicious tag that contains a hardware Trojan. Finally, RFIM platform helps to develop countermeasures against the fault effects. These countermeasures have been implemented and tested in a real RFID environment with several tags and reader.

Keywords
RFID, FPGA, Fault injection, Hardware security, Prototyping, hardware robustness

Contents

List of Figures:

List of Tables:

Chapter 1

1 Introduction

UHF RFID tags are increasingly used for critical applications within harsh environments (aeronautics, railways) or for secure applications such as identification, countermeasure against counterfeiting. However, such low cost systems, initially designed for non-critical applications with a high production volume, are not robust by themselves. For critical applications, a malfunction of the RFID chip may have serious consequences or may induce a severe security breach for hackers. Dysfunctions can have many origins: for instance, hardware issues can be due to aging effects or can also be due to hacker attacks such as optical or electromagnetic fault attacks. It is thus a common practice for critical applications to increase the robustness of RFID systems. The main purpose of this PhD thesis is to increase UHF tag robustness by first proposing a dedicated robustness evaluation platform and then proposing new digital architectures of UHF RFID chips which would be resilient against both hardware attacks and natural defects.

Classical design techniques for robustness IC improvement aim to evaluate the design robustness by simulation with fault injection. Within the RFID context such an approach only based on simulation has several drawbacks. In fact, simulations are often inaccurate because the RFID system are heterogeneous and their behavior relies on several parameters such as the global electromagnetic environment, the number of tags present in the reader field, the RFID protocol parameters, etc.

The purposes of this thesis are to develop validation methods and tools dedicated to RFID system in order to avoid time consuming simulations and then to evaluate the design within a real environment.

In this thesis, an FPGA based emulation platform called RFIM is presented. This platform is compliant to EPC C1 Gen2 RFID standard [EPC2008]. The RFID tag emulator has been validated functionally in a real environment. The RFIM platform uses the instrumentation fault injection technique to inject faults in the digital tag circuit. Thanks to fault injection campaigns, RFIM platform allows analyzing the effects of the faults injected into the tag, and then helps designer validating new robust digital architectures within a real RFID environment. The RFIM platform can be used to demonstrate the effects of malicious attacks against RFID systems based on fault injection or malicious tag modification via Hardware Trojan.

Chapter 2 of this thesis is devoted to a presentation of the general concepts of RFID technologies: classification, regulations and RFID standards. We show the importance and the benefits of using RFID technology, especially in critical applications, which demand a high level of security and dependability. Following this presentation, we discuss about the heterogeneity and complexity of RFID systems. These complexity and heterogeneity involve that their dependability improvement is a difficult task. In this chapter, we finally justify the need for a validation platform that takes into account all the RFID system parameters.

In chapter 3, we present a state of the art of different RFID system validation platforms and a brief description of existing robustness evaluation tools. In the first part of this chapter, we detail the RFID system simulation based platforms and then we give more detailed descriptions on RFID system emulation platforms. In the second part of this chapter, we discuss the methodology of the fault injection. Then, we conclude this chapter with a state of art of different fault injection platforms that are proposed in literature. We distinguish between two categories of fault injection platforms: one based only on simulation and other based on emulation or prototyping.

In chapter 4, we present our RFID tag emulator (RFIM) with fault injection capabilities to emulate errors in order to validate the robustness of both tag architectures and its protocol. Using RFIM, in system fault injection is carried out and the results are presented and detailed. Finally, some architecture enhancements in order to increase the tag robustness are presented and evaluated.

In chapter 5, we focus on the RFID vulnerability to Hardware Trojan by presenting general research results on HT insertion and detection. Then, we show how the hardware emulation platform RFIM can help to evaluate some trigger and payload of Hardware Trojan. Finally, we discuss some countermeasures which could be designed at the system level in order to protect RFID system against untrusted RFID tags.

Finally the conclusion summarizes the main results of this work, and discusses perspectives.

Chapter 2

2 Safety and RFID Systems

2.1 Introduction

In this chapter, we will present RFID systems from technology issues to usage and standard issues. We will first provide important information on RFID technology which are necessary to set-up the context of this work. Then, we will show the importance and the benefits of using RFID technology, especially in critical applications, which demand a high level of security and dependability. RFID systems are complex and heterogeneous systems. As a result to this complexity and this heterogeneity, analyzing and improving the dependability characteristics of such a system is a complex task. In this chapter, we will justify the need for a validation platform which deals with all the RFID system elements.

2.2 RFID technology

RFID (Radio Frequency Identification) is an automatic identification technology that uses radio frequency waves to identify objects attached to a tag when they pass close to an RFID reader. When communication is established between a tag and a reader, the data contained in the tag are transferred to the reader. The data in the tag can also be modified with specific commands issued by the reader. The radiated electromagnetic waves transmitted by the reader include information and at the same time provide energy to the tag which allows this one to process data before to backscatter information to the reader.

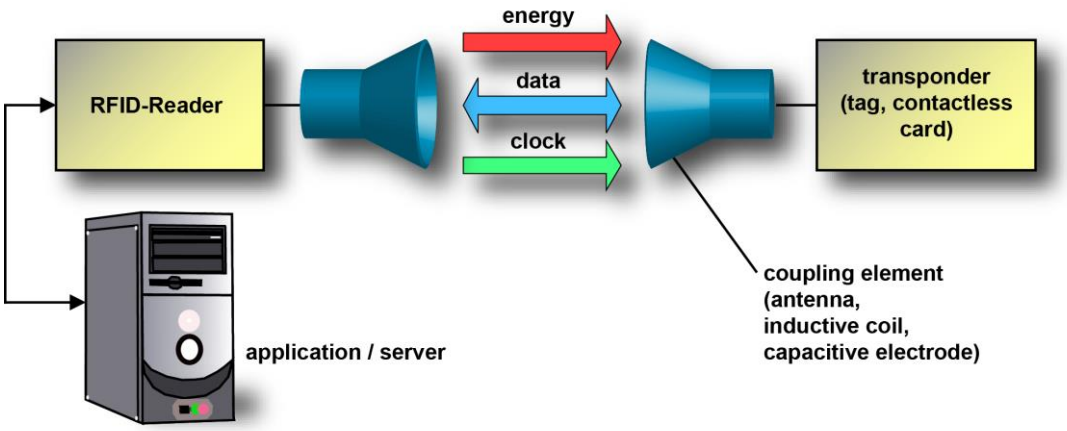


Figure 2.1: RFID system operations

Figure 2.1 describes the general operations of an RFID system. We can summarize the communication between a tag and a reader as follows:

- The server must be able to handle the reader and to generate appropriate commands.

- Communication between the reader and the tag is done by radio frequency signals.
- The signal carrying command information is generated by the reader.
- The carrier signal is emitted from the reader antenna to the tag.
- The tag receives the transmitted signal, extracts digital information and then transmits the response to the reader
- The reader antenna receives the modulated signal from the tag, and then demodulates it.
- The reader decodes the received signal and extracts digital information.
- The host server retrieves the received information; this information can be filtered, modified or aggregated before being stored in the database of the host.

2.2.1 *RFID technologies*

2.2.1.1 *Frequencies used in RFID Technologies*

Since RFID technologies use radio waves for communication between tag and reader, these technologies are then classified by frequency bands in which they operate. The different frequencies are allocated by regulatory authorities who set very specific rules to use each frequency. RFID systems work from Low frequency (LF) to Super High frequency (SHF). These frequencies have been standardized to avoid interferences from other devices of other technologies. All frequencies used for RFID technology are summarized in Figure 2.2.

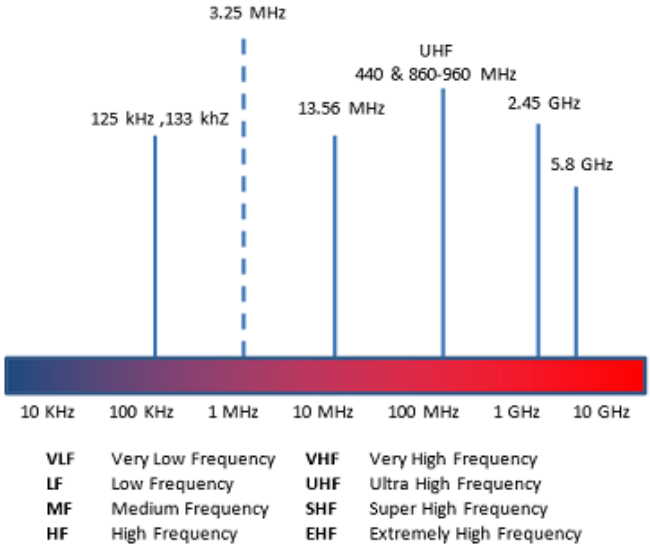


Figure 2.2: RFID Frequencies

a) **Low frequency**

From 100 kHz to 500 kHz, the reading distance is a few centimeters; these frequencies are used in industrial environments as well as for animal tracking. This band allows reading in any environments but only for short distances (less than 10 centimeters).

b) Medium frequency

From 10 MHz to 15 MHz with a reading distance of 50 to 80 cm, these frequencies are particularly used in library logistics, flows monitoring and access control. This band allows a medium reading distance (about one meter). Tags using this frequency range are more sensitive to nearby presence of metal or liquid.

c) High frequency

From UHF band (850-950 MHz) to SHF band (2.4 to 5.8 GHz), the reading distance is several meters (in free air, this distance can be reduced by the presence of metal or liquid). These frequencies are particularly used in supply chain management.

2.2.1.2 *RFID tag types*

RFID tag can also be classified according to their power supply source. We distinguish three different power supply types:

a) Active tag

This type contains an embedded battery allowing them to emit a signal without needing to be remotely powered. This type of tag can achieve reading distances of a few meters. They are mainly used to send big amounts of information over long distances. They have the disadvantage of being more expensive than other systems; they require maintenance service and are less integrated.

b) Semi-passive tag

Semi-passive tags, also called semi-active, are very similar to the active tags since they are also powered by an embedded energy source (Figure 2.3). The difference between these two types of tag is related to the battery used. The power of the semi-active RFID chip is not used to send any signal but for storing data between successive communication exchanges. The semi-active tags use backscattering techniques to send back the reader signal.

These tags are often used to record data during goods transporting. They are particularly useful in the field of food traceability and logistic traceability that require to:

- Register temperature changes during transport,
- Surveillance of equipment fleets, etc.

Figure 2.3: Semi passive RFID tag example

c) Passive tag

Passive RFID tags are the most deployed because of their low cost, miniature size, and the simplicity of their architecture. They do not include neither battery nor RF transceiver as the semi-passive and the active tags do. This type of tag use the backscattering technique to response to reader commands. The backscattering technique consists in modulating the scattered electromagnetic wave incident from the reader. The scattered wave is modulated by changing the electrical impedance of the tag antenna. A passive backscatter tag receives the power needed to operate from the wave incident from the reader. This type of tag is compact and simple. The passive tag is composed of an antenna for communicating with the reader and an electronic chip which integrates memories that can store data to be transmitted. All these elements are combined in a packaging as shown in Figure 2.4

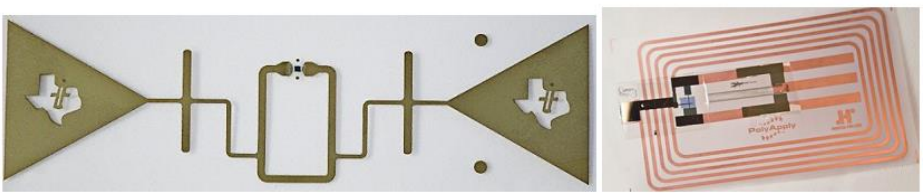


Figure 2.4: Example of passive UHF RFID tag

2.3 RFID system components

In this work we focus on passive tags. Indeed, thanks to its low cost, passive UHF RFID technology remains the most deployed one for traceability applications.

2.3.1 *Passive UHF RFID tag architecture*

A passive UHF RFID tag consists of an antenna and an integrated circuit that includes both the radio frequency part (Analog front end), and the digital part (digital baseband). An overview diagram of an UHF tag is shown in Figure 2.5.

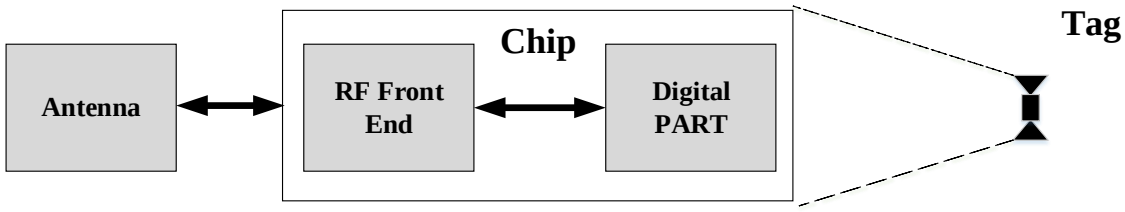


Figure 2.5: Block diagram of a UHF RFID tag

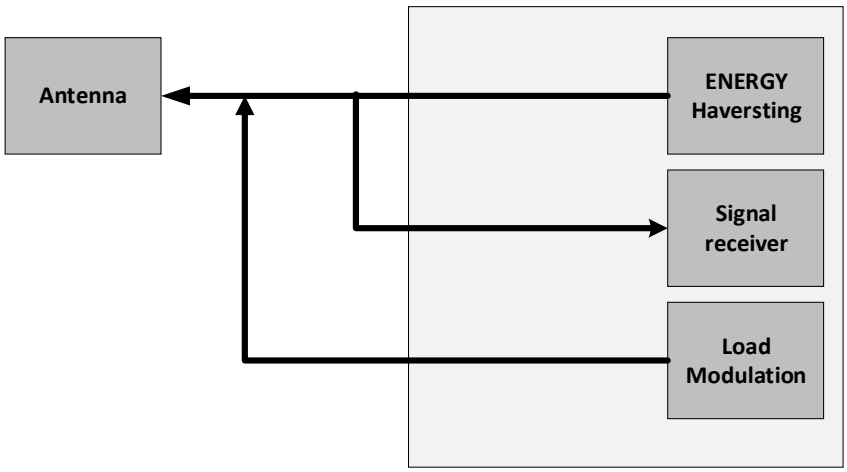


Figure 2.6: Detailed Block diagram of a UHF RFID tag

The different features of the radio frequency front-end, as shown in Figure 2.6, can be described as follows:

- The energy recovery function is performed by a rectifier which can recover a DC voltage from the RF carrier sensed by the antenna. This voltage is supposed to power the tag.
- Frequently, the rectifier is connected to a regulator or a voltage limiter allowing the voltage stability and protecting the chip from overvoltage risks.

- All requests transmitted by the reader are provided to the signal receiver block. This block contains a demodulator, a baseband filter and an analog to digital converter. The demodulation type used in UHF RFID tags is the ASK (Amplitude Shift Keying).
- Finally, the backscattering function is performed by the load modulation block which modifies the load at the input of the antenna.

We present in Figure 2.7 a block diagram illustrating in more details the complete UHF RFID tag architecture. The radio frequency part (RF Front-End), contains ASK demodulator, a rectifier, and load modulation part.

The digital baseband contains the power management block to ensure a stable energy supply to the tag and the baseband signal processing block used to encode and to decode the digital baseband signal. The role of the modulation control block is to control the load modulation part to generate the backscattering signal. Finally, the digital part manages the execution of the RFID protocol [EPC2008].

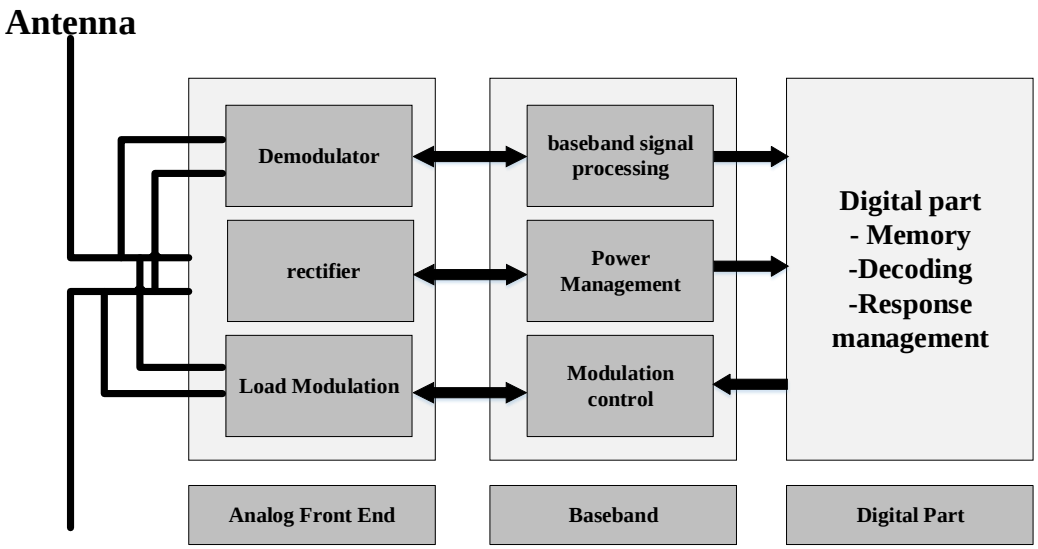


Figure 2.7: Block diagram of UHF RFID tag architecture

2.3.2 *UHF RFID reader architecture*

The RFID reader is the interface between the application/server and the RFID tags as shown in Figure 2.1. Executed by the server, the middleware is a software which is used to filter the tags data and manage several readers across a network. The reader just ensures the transfer of all information extracted from the tags to send it to the middleware.

The main role of the reader is the communication establishment: it aims at managing the communication with the RFID tags and with the server. It executes the middleware to communicate with the server and controls the communication protocols to communicate with tags. For example, for UHF RFID, one standard protocol is the EPC C1 Gen2 [EPC2008]. This protocol will be described in the section 2.4.1. All these functions are performed by a digital part (as shown in Figure 2.8), composed of microprocessor, memory and an RF interface controller.

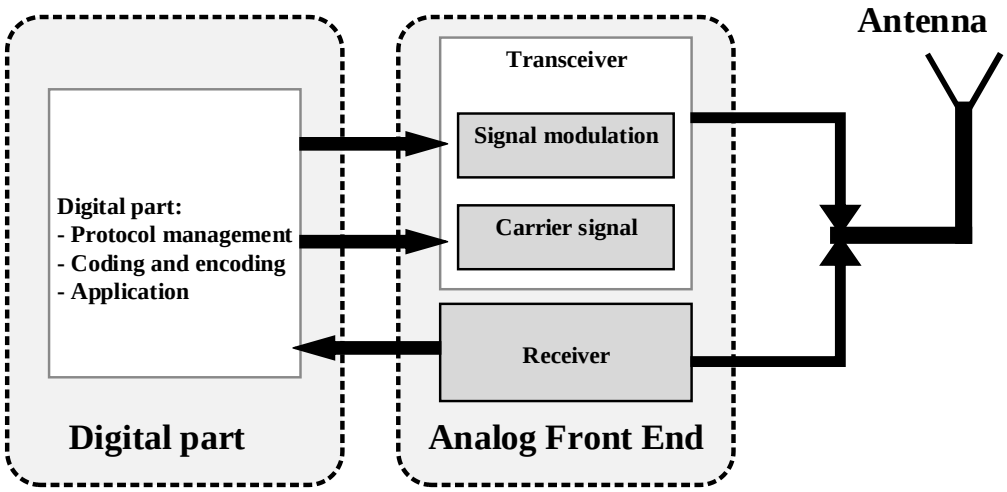


Figure 2.8: Block diagram of UHF RFID reader

All RFID reader features are summarized in the block diagram of Figure 2.8. The digital part ensures the digital signal generation. This signal contains information to be transmitted to the tag. The digital part also manages information returned by the tag. In addition to the implementation of the communication protocol, the digital part is able to encode and to decode signals and optionally to encrypt and to decrypt information, or to perform any other functions required by the application (middleware).

As shown in Figure 2.8 the physical layer of the communication is realized by a Radio Frequency Front-End which consists of a transmitter and a receiver. This RF front-end is thus responsible for generating carrier frequency to remotely supply the tag.

2.3.3 *RFID applications*

Applications based on RFID systems are more and more numerous. We will not describe in this document all the possible applications. Since we are interested by critical systems, we will only focus on critical applications that use RFID technology: security, health and military applications.

a) Health

In the health sector, RFID technology is used in some applications such as:

- **Hospital Management:** Equipment available in hospitals can contain tags for tracing their cleaning, disinfection, sterilization and availability. Another very interesting application is to ensure the traceability of blood donation through a system that contains a temperature sensor allowing permanent control along the reservation chain.
- **Medical Surveillance:** RFID bracelets can be attributed to patients, replacing the care sheets. They contain the number of the doctor or of the nurse, and are connected to a database that stores all the patient records. The system can be accessed via tablet or PCs.

Securing medical information stored in the RFID tag is an important issue. Indeed, integrity and availability of medical sensitive information must be guaranteed.

b) Security:

Many security applications are also using RFID technologies, among them we can give the following ones.

- **Passport:** This is an important example of RFID technology usage in security applications. All individual information are stored in RFID tag and the RFID technology is used to check the validity of the documents.

c) Military application:

The US military has been one of the early adopters of RFID technology. They have used RFID for over ten years in a limited area of their operations. In 2003, they improved their use of this technology by requiring that all suppliers must put an RFID tag to each pallet or product being shipped to the military. The American Department of Defense (DoD) estimated that the military has saved 300 million US$ in Iraq by using RFID [Silicon2005]. In fact, this technology has played a very important role in logistics for military operations through a fully automated and visible management resources. This has allowed commanders to visualize in real time the movements of all war materials.

In France, the French Ministry of Defense launched a new IT project called System Monitoring Logistics Information of the Inter-Armed Resources [Capgemini2013]. The

objective is to follow the routing of all French army materials of all the French forces in the world. All materials of the French army will be equipped with RFID tags.

All these applications and examples show the pervasiveness of this technology in our life as shown in Figure 2.9 which illustrates all the applications using RFID.

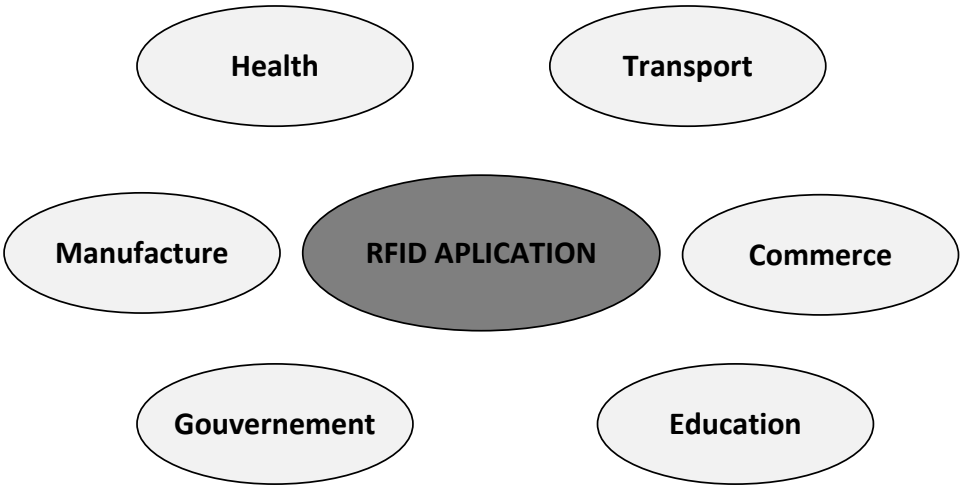


Figure 2.9: Application areas of RFID technology

After presenting the interests and benefits of RFID systems, we will spread out on different communication protocols between components constituting the RFID systems and their architectures. In this thesis, we focus on UHF RFID technology that is compliant with the EPC C1 Gen2 standard [EPC2008]. This technology is widely used in many applications because it ensures the data transfer over several meters, and with data throughput higher than for other RFID technologies (allowing reading more than 200 tags at a time). In order to understand the different elements which, fulfill this standard, we first give a description of this one.

2.4 EPC global (Electronic Product Code)

In this work, an RFID tag compliant to the EPC Global Class 1 Generation 2 (EPC C1 Gen2) standard has been designed. This tag will be used to evaluate RFID system dependability using fault injection in its architecture. In order to understand the functionality of this RFID tag, it is important to know this EPC C1 Gen2 standard. Understanding this standard will also allow identifying dependability issues and defining potential countermeasures.

EPCglobal is the organization that leads the development of standards for the EPC to support the use of RFID.

EPC C1 Gen 2 defines the physical and logical requirements for a passive-backscatter tag and Reader-Interrogator operating in the 860 MHz ~ 960 MHz frequency range. Both RFID tag and reader physical and logical requirements are defined in EPC C1 Gen2 **[EPC2008].** In this standard two layers are introduced: the Physical Layer and the tag identification layer.

2.4.1 *EPC C1 Gen2 features*

The most important EPC C1 Gen2 protocol functionalities are tag inventory feature, which differentiates it from other RFID standards. This feature is based on four commands which are:

1. **Select**
2. **Query**
3. **Query Rep**
4. **QueryAdjust**

These commands are used to read the EPC number of tags which are into the reading range of UHF RFID reader. This identification number is used to identify the tag holder thanks to a database that stores information about this tag in a server. The inventory commands allow inventorying all the tags found in the field of the reader in a very short time. The standard also defines the commands allowing reading and writing within the tag memory. These commands are the so called access commands.

2.4.1.1 *Physical Layer Communication Features*

In EPC C1 Gen2 protocol, the reader controls all communication features and tunes most of the parameters in physical communication layer including the two communication links:

- Reader-to-tag link
- tag-to-reader link

These two links are different. Each one has its own data rates, data encoding schemes and parameters. This allows readers to adapt them at any environment change by setting specific parameters for these two communication links.

For example, if the environment is very noisy, it is preferable to use the FM0 encoding that is more immune against noise and error communication. Another example concerns the reading of a large number of tags. In this case, in order to speed up the read rate, the reader changes inventory parameters to get the fastest data read rate. The different data rate options and types will be detailed in the following section.

2.4.1.2 *Data encoding*

The EPC C1 Gen2 protocol reader-to-tag link uses PIE (Pulse Interval Encoding) code [EPC2008]. In this encoding scheme two pulses of different length represent data 1 and data 0. Data 0 pulse duration is shorter than the data 1 pulse duration. Data 1 is represented by a low level for a short time followed by a high level for a long time. Data 0 is represented by high and low level of the same periods as shown in Figure 2.10

Figure 2.10: Pulse-Interval Encoding (PIE) Baseband Symbols

Tari is the reference time interval for EPC C1 Gen2 standard. It can range between 6.5 us up to 25 us. Data 0 duration equals Tari length. The high signal values represent transmitted continuous wave (CW), and the low signal values represent attenuated CW as shown in Figure 2.11.

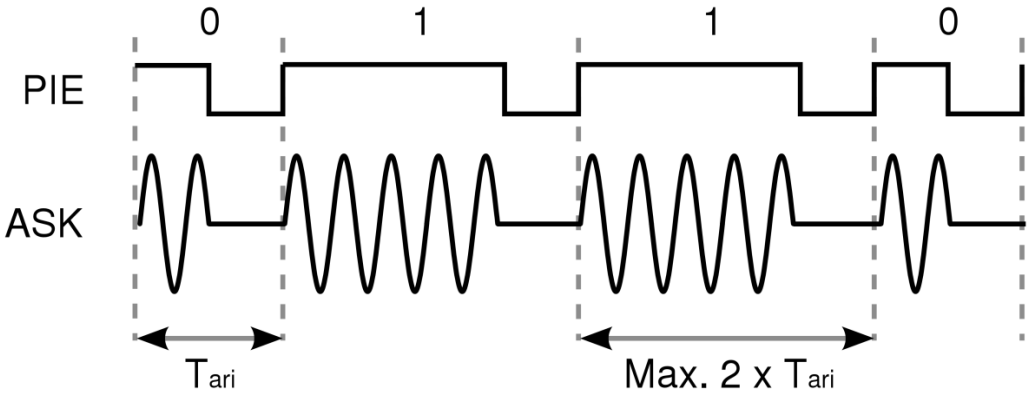


Figure 2.11: Data encoding in PIE format.

In EPC C1 Gen2 standard, the reader adds at the beginning of the communication some timing information to each packet it sends. These informations specify the reader and tag communication rates. These timing information can be included in a Preamble or a Frame-Sync. A preamble is added at the beginning of a Query commands, and a Frame-Sync is added at the beginning of all other commands. The difference between a Preamble and Frame-Sync is the inclusion of TRcal which gives the backscatter link frequency (BLF). Figure 2.12 and Figure 2.13 describe these two types of beginning frame.

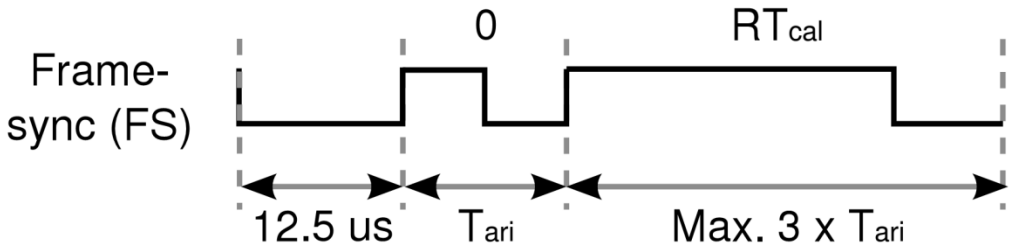


Figure 2.12: Reader-to-tag frame sync.

The preamble contains only four components: (1), the delimiter; (2) data 0 bit; (3) the reader-to-tag calibration RTcal symbol; and (4) the tag-to-reader calibration TRcal symbol [EPC 2008], as shown in Figure 2.13.

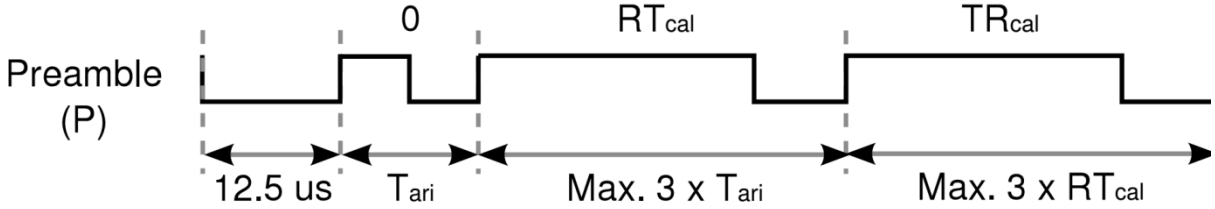


Figure 2.13: Reader-to-tag preamble.

TRcal holds information about data rate and data encoding schemes that will be used by the tag during response to reader commands. During *Query* command decoding frame, the tag parses a parameter called *divide ratio* (*DR*), and using *TRcal* symbol duration tag can compute data rate for tag to reader link. The data rate is calculated based on the BLF (backscatter link frequency) value [EPC 2008] using the following equation **[EPC 2008]:**

$$\mathbf{BLF} = \frac{\mathbf{DR}}{\mathbf{TRCAL}} \quad \textbf{Equation 2-1}$$

We note that BLF is calculated only one time at the beginning of every inventory round. This is done when the *Query* command is received by the tag. Then it's fixed until the end of the current inventory round.

2.4.1.3 *Miller and FM0 encodings: tag to Reader link*

The encoding methods used can be FM0 Baseband (40-640 Kbit /s) or Miller Sub-Carrier (5-320 Kbit / s). In FM0, encoding transitions (from 1 to 0 or 0 to 1) must be at the end of each bit period and additional bit 0 at half the bit period transition is necessary. This is shown in Figure 2.14.

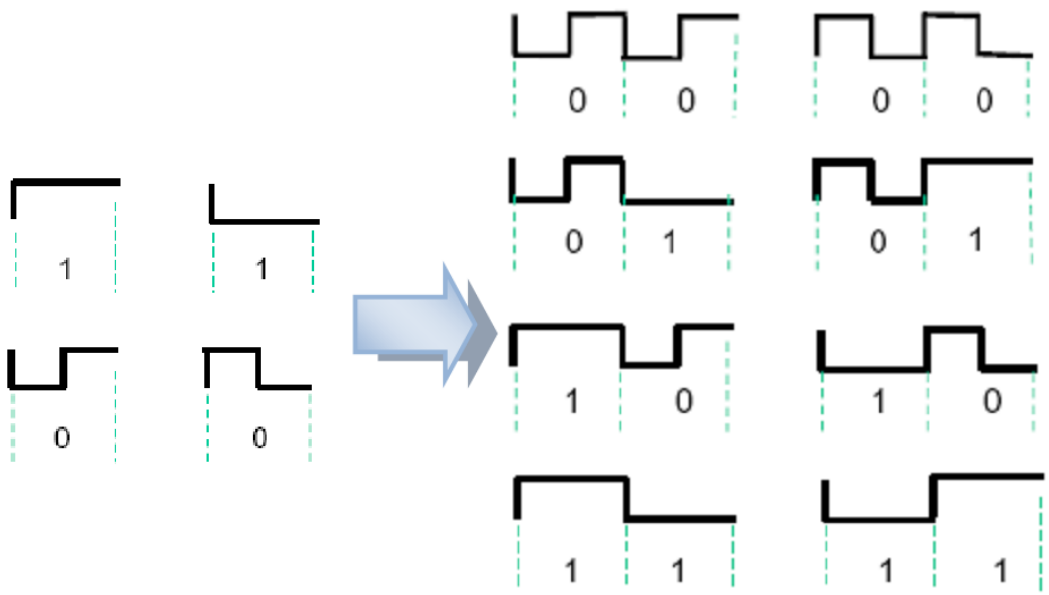


Figure 2.14: FM0 encoding the tag to reader link. On the left), representation of bits 0 and 1. On the right a few possible sequences [EPC2008]

Miller encoding is characterized by the transition that occurs between two zero or two one sequences. As shown in Figure 2.15, a sequence of Miller may have M = 2, 4 or 8 cycles per bit. The M value is present in the query command frame. This parameter is defined by the reader at the beginning of the inventory round.

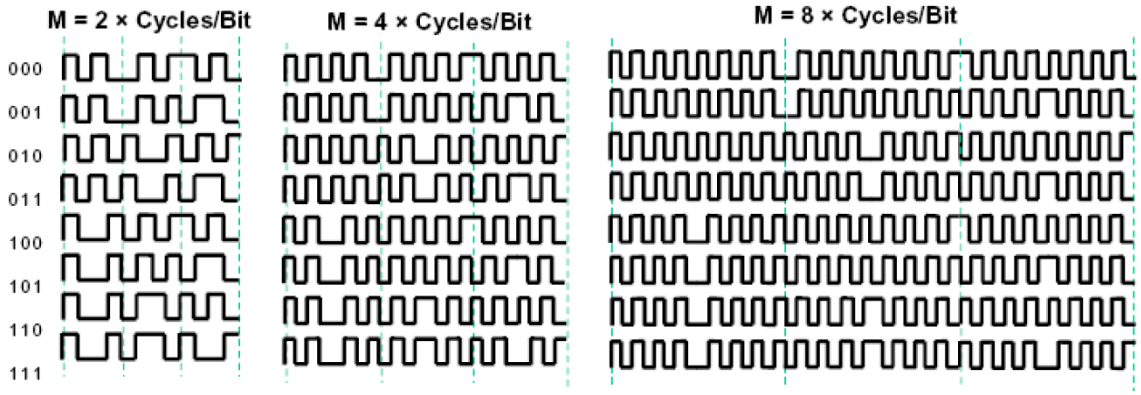


Figure 2.15: Miller encoding schemes

Figure 2.16 shows how to use Miller encoding, as well as two possible modulations to use this link: ASK or PSK.

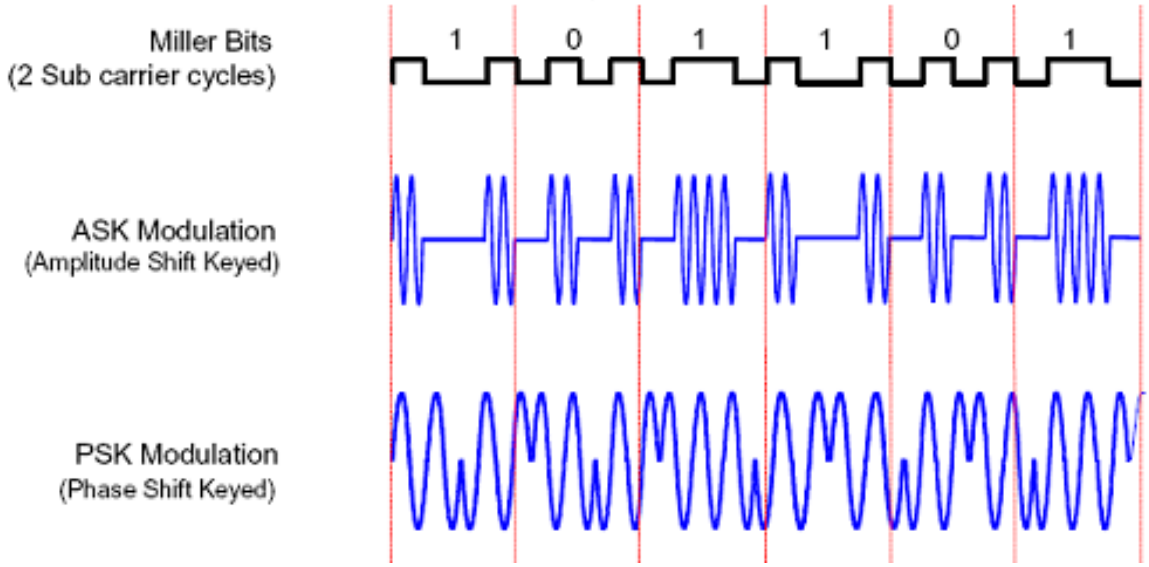


Figure 2.16: tag to reader Miller encoding

2.4.1.4 *Sessions & inventoried flags*

If more than one reader tries to communicate with a single tag at the same time, consistency problems will occur. In this case to allow multiple readers to communicate with a single tag EPC C1 Gen2 standard provides solution support for allowing up to four readers to communicate with the same tag in the same time. This mechanism is called sessions and is based on inventoried flags.

Sessions are useful because they can be used to set the inventoried flags of all tags to a known value. After reading the EPC number of each tag the inventoried flag will be switched. Tag will no longer participate in the current inventory round of this reader. This allows it to communicate with other readers using different sessions.

When the tag is powered on, session flag S0 is set to A, and the other session flags S1 to S3 are set to their previous values (A or B). The SL flag (select flag) is used by the reader to select a tag population before starting an inventory round. The tag modifies its SL flag only when the reader sends the select command before inventory round [EPC 2008].

2.4.1.5 *Tag memory*

The tags have 4 banks of non-volatile memory as shown in Figure 2.17.

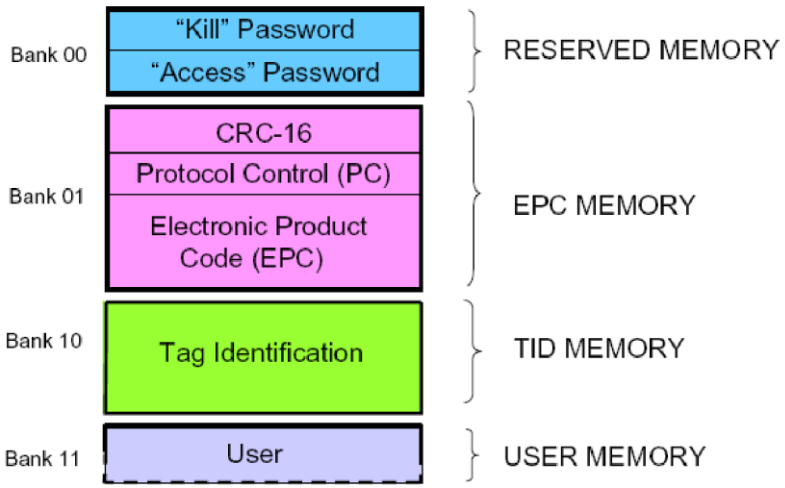


Figure 2.17: Memory banks of a tag [EPC2008].

1. **Reserved Memory:** contains two 32 bit passwords, one to access to tag and the second to kill the tag. Kill operation permanently disables the tag. The access password allows tag to enter into a state called the secured state.
2. **EPC Memory:** contains the Cyclic Redundancy Code (CRC), Protocol Control (PC) that contain physical-layer information, and EPC code (Electronic Product Code). During an inventory round the tag backscatters these three codes to the reader.
3. **TID Memory:** contains the tag identifier.
4. **User Memory:** this bank is optional and contains specific data.

2.4.1.6 *Commands and state machine*

There are three basic operations that manage the tag behavior, as shown in Figure 2.18.

1. **Select:** this command determines which group of tag will respond, i.e. which tags will answer in the next state.
2. **Inventory**: identification of tags; in this phase, the reader identifies all tags by their identifier stored in their EPC memory.
3. **Access**: once the tags have been identified, reader can exchange data with the tags by reading and writing data in their memory.

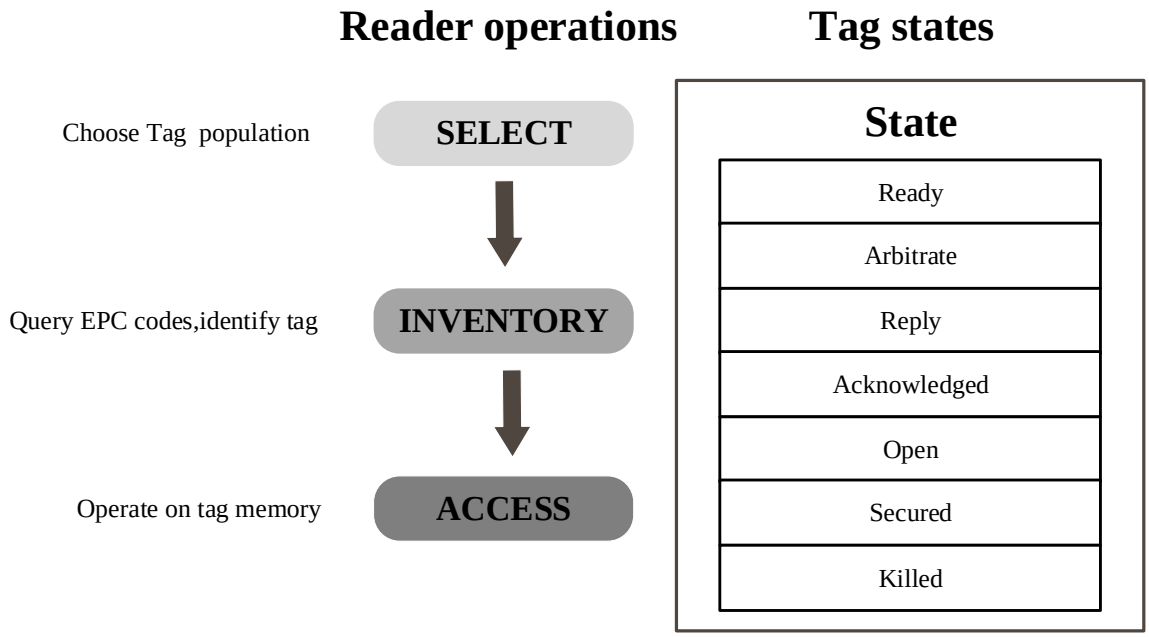


Figure 2.18: Commands and state machine

Slot counter: at the beginning of the inventory round the reader sends select command then query command. The tag picks up the Q field in query command frame to generate a random number to preload their slot counter, then every time the tag receive a query rep, it decrements the counter, and when reaching the value of 0 the Tag responds with a 16bit random number.

All operations according to EPC C1 Gen 2 protocol follow the state diagram described in the air-interface specifications in the section 6.3.2.4 of [EPC 2008]. A simplified version of this state diagram following the reader operations is summarized in Figure 2.19.

At the beginning of the inventory round the reader sends the select command then the query command. The tag picks up the Q field found in the query command frame to generate a random number to preload a counter (named slot counter). Then, every time the tag receives a query rep command, it decrements the slot counter. When this counter reaches 0, the Tag answers with a 16 bits random number (RN16_1). This mechanism allows avoiding collision when multiple tags try to communicate simultaneously. Then the following commands sent by the reader will allow to obtain the EPC number and to access to the tag memory.

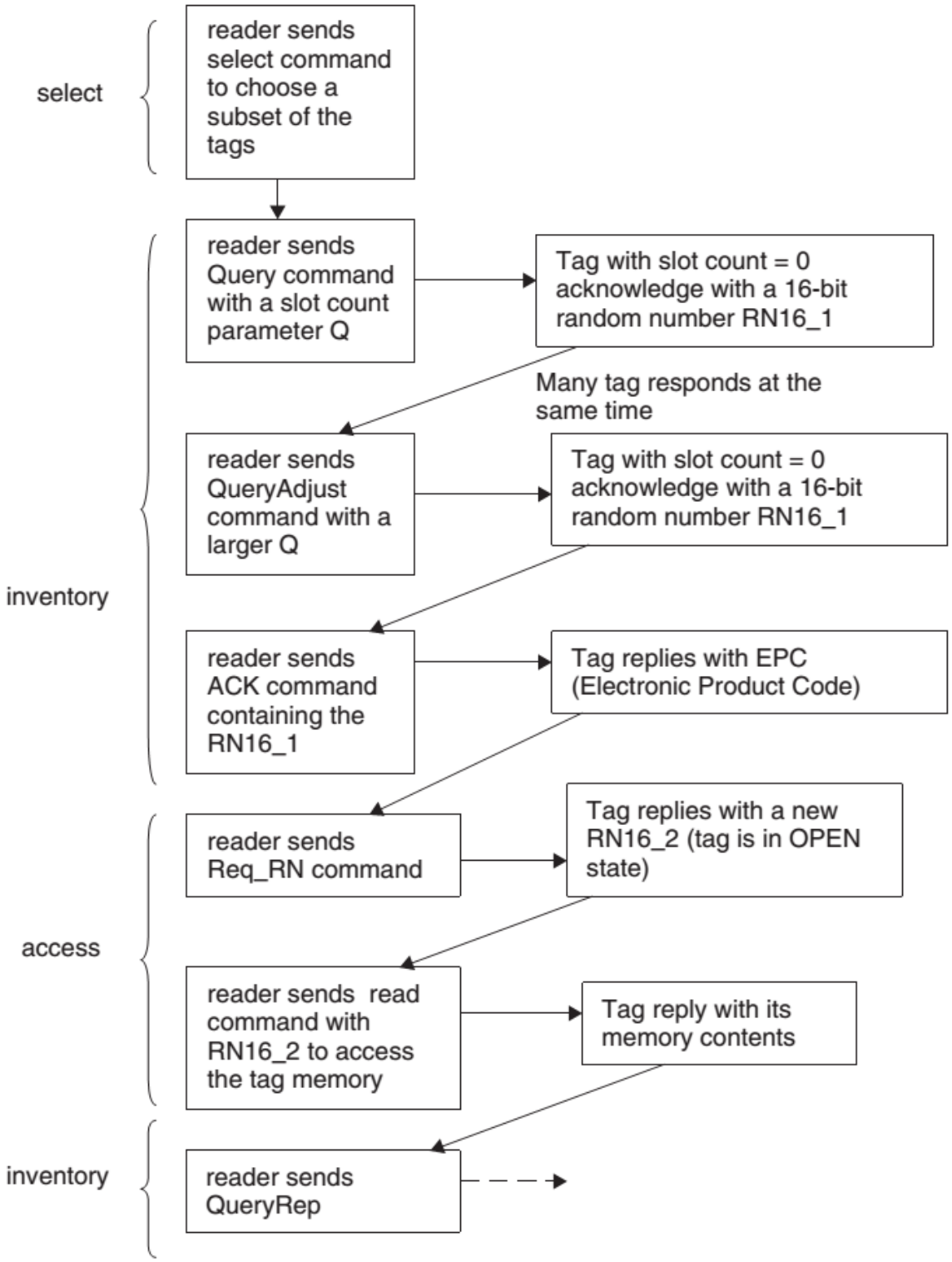


Figure 2.19: Simplified protocol followed between a reader and a tag [EPC2008]

The tag behavior can be described by a finite state machine; this state machine has a large number of states that depend on several parameters, such as slot counter, flags, current state... Using tag state diagram shown in Figure 2.20, we briefly introduce the meanings of each state and transition rules as described in [EPC2008].

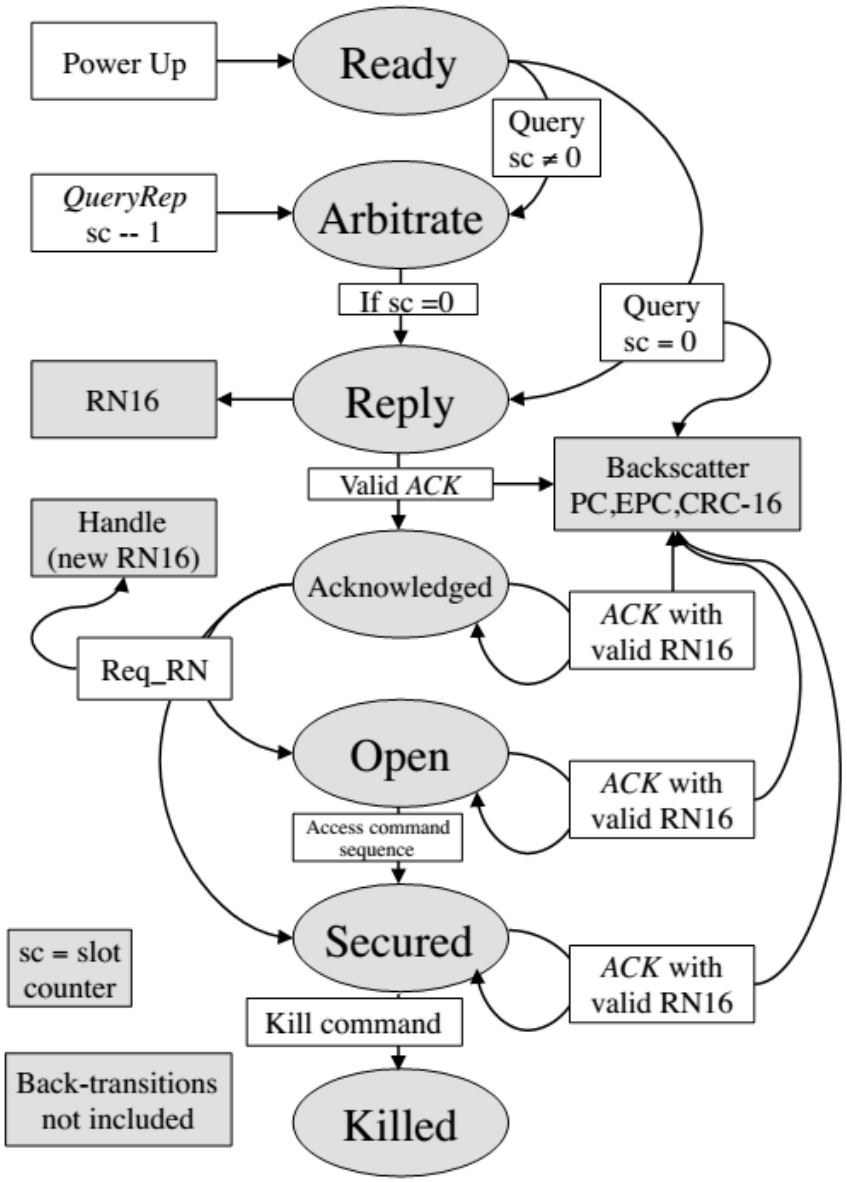


Figure 2.20 : Tag state diagram [EPC2008]

1. **Ready state:**

The initial state when the tag is power up is ready state except in the case the tag is killed.

2. **Arbitrate State:**

Tags in this state participate in the inventory rounds, and load their slot counter with a random value. If the slot counter is different from zero, the tag still in arbitrate state and counts down this value until it reaches zero. Then the tags go to the reply state. If the tags load the initial slot counter with a zero value, will directly answer to the reader.

3. Reply state:

The tag in this state answers with a 16-bit random number.

4. Acknowledged State:

When the tag is in reply state and the reader sends an Acknowledged command to confirm their response. The tag then transmits its EPC number followed with Identifier called program counter (PC) and CRC that protect the sent frame. The tag remains in this state for a time equals to T1 [EPC2008]. In the absence of any new command from reader, tag will automatically return to arbitrate state. In this state that the reader is able to transition the tag to the Open (or Secured) state allowing access operations such as Read, Write, Lock, and KILL

5. Open state:

When the tag is in acknowledge and the reader sends a request random number (Req_RN) command. If the password stored in reserved memory of the tag equal zero then the tag goes directly to secure state, and if the stored password is different from zero then tag go to open state, and there will be some challenge response between the reader and the tag before the tag go to secure state, this communication is shown in Figure2.21.

The tag replies by a new 16-bit random number that is called Handle step2, any command requested by the reader must include this random number as a parameter in the command. The reader provides 32-bit password that protects the writing operation. This password consists of two 16-bit sequences, denoted in Figure2.21 as Pw2 and Pw2 code. The reader gets in steps 4 and 8, two random sequences of 16 bits, denoted as RN16 'and RN16". In step 5, the reader applies an XOR operation between the first 16-bit word sequence and the RN16’. Then, it sends the result to the tag for a tag response in step 6, then the reader applies an XOR operation of the remaining 16 bits of the password and the RN16” sequence ", and sends the result to the tag in step 9. The tag recognizes the reception in step 10 by sending a new handle to the reader. Using the latter, the reader can do all access command such as read write kill operation reader step 11.

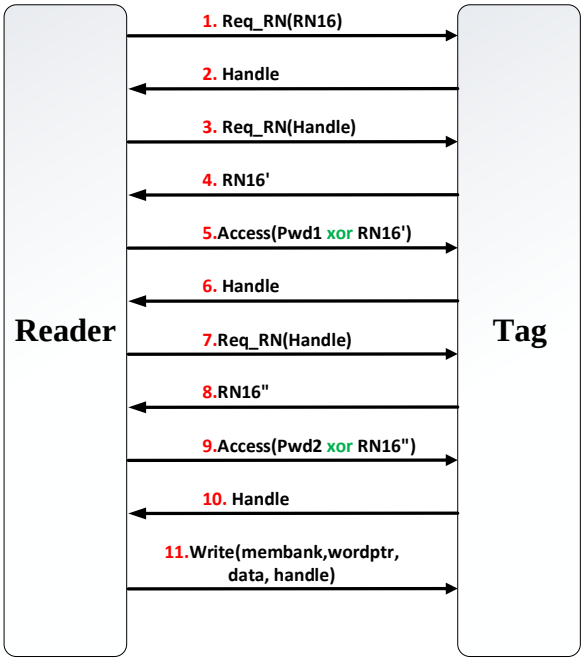


Figure 2.21: Access protocol of an EPC C1 Gen2 tag

6. Secured state:

The tag transits to secured state after receiving a *ReqRN* and the Access password is zero. If the tag implements a password, the reader must provide the correct password as described in open state subsection.

7. Kill state:

If the tag receives the kill command with a valid Kill password while it is in the secured state, then it will transit to kill state. The tag will stop any response to reader request and will never be reused.

2.5 Safety and security validation of UHF RFID systems

As we have seen the use of RFID has considerably increased in several critical areas as medical, industrial or military applications. This puts tremendous pressure on designers of RFID systems for security and reliability of critical applications.

Eventual design problems or weaknesses not detected at design time can lead to serious materials or financial damages. Therefore, it is important to evaluate RFID system robustness at design time in order to provide safe and secure systems. The evaluation should tackle both natural and malicious faults which can decrease the system robustness. Due to the inherent heterogeneity and complexity of such systems discussed hereafter, such validation at design time can hardly be performed considering the whole system.

The aim of this work is to present a novel system evaluation platform using emulation and fault injection techniques to provide an additional tool for developers of RFID systems. First, such a tool aims at validating tag hardware architecture taking into account the entire RFID system component interactions. Secondly, it aims at enhancing and validating security-and reliability of the tag once again taking into account the whole RFID system. Faults are emulated using flexible fault injection method. The platform developed in this work enables the efficient evaluation of new hardware implementations of critical security or reliability solutions for UHF RFID tags.

2.5.1 *Safety issues*

RFID tags are increasingly used into critical applications within harsh environments or for secure applications such identification, counterfeiting protection, etc. However, such low cost systems, initially designed for non-critical applications with a high volume, are not robust by themselves. The behavior of a RFID system must be the same than the one described in the functional specifications, in terms of both performances and functions. The functions are described by the sequences of states of the EPC C1 Gen2 standard **[EPC2008].** In these states, the RFID tag performs computation, communication, and store information. The service delivered by the tag is a sequence of information in the form of commands and information requested by Reader. To deliver a correct service regardless its environment, a tag must guarantee the following attributes:

- **Reliability:**

The probability that the tag gives correct outputs within a given time interval $[t_0,t_1]$ **[Johnson1989]**. For this attribute, our objective is to ensure that the tag response is always furnished during an inventory round even in harsh environments or in the presence of attacks.

- **Availability:**

The conditional probability that the tag will correctly work within the time interval [t0, t1], given the system was performing correctly at time t0 [Johnson1989]. It means the tag is ready to be used.

- **Safety:**

The probability that an RFID tag will either perform its functions correctly or will discontinue its functions in a manner that does not disrupt the operation of other tags found in the field of

reader, [Johnson1989]. This is related to the nonoccurrence of catastrophic consequences on the environment.

- **Integrity:**

The absence of improper system alterations, which can appear for example using a malicious reader [Johnson1989].

- **Maintainability:**

The ability to undergo modifications and repairs these modifications by introducing a mechanism of fault tolerance **[Johnson 1989].**

We will consider an UHF tag delivers a correct service when the service accurately reflects the EPC C1 Gen 2 function requirements. Failures are either because the tag is not well conforming to the standard [EPC2008] or because its architecture does not adequately describe a robust system. In this thesis we will focus on the reliability and the security of tag digital part. We will describe in the following the potential issues in the reliability of digital part of RFID UHF tag.

Since the tag functionalities are based on a correct sequence of states, if a fault occurs then the tag can deviate from its correct state to another. This deviation is called Error. This error can propagate and be transformed to another failures as shown in Figure 2.22 (fault, error and then failure from component A propagates to component B). In Chapter 4 we will analyze in details all fault propagation mechanisms available in the tag architecture.

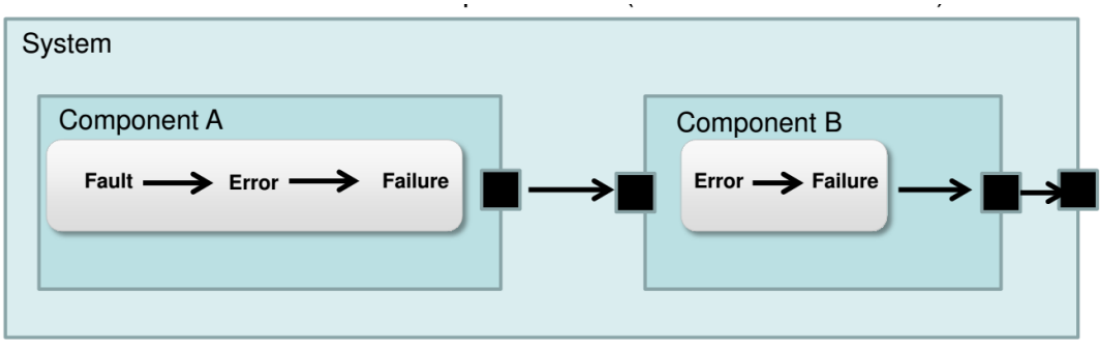


Figure 2.22: Error propagation in digital system

Figure 2.23 shows the relationship between faults, errors, and failures [Johnson 89]. When a fault occurs, it can cause errors, and errors may propagate to cause service failures.

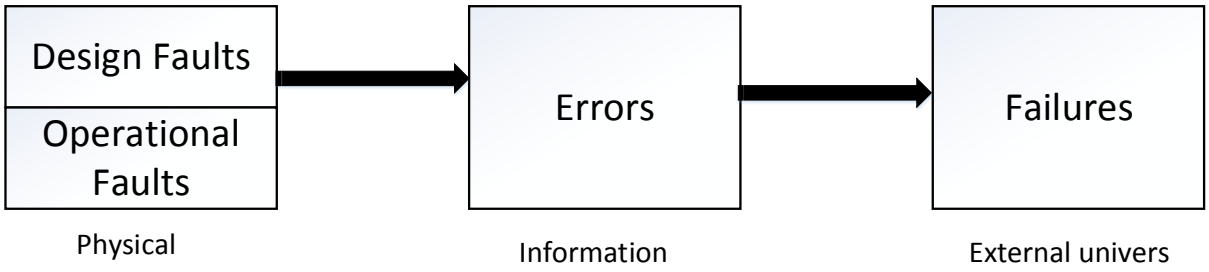


Figure 2.23: Relationship among faults, errors, and failures

We explain this relationship more in details with an example in the context of an RFID tag. When a tag makes an incorrect state transition, this can be caused by a fault in its design or by a bit flip in one or more registers. This fault can involve a wrong transition of the tag. For example, a transition from the ready state directly to the reply state without passing by the acknowledge state. We can say this is the failure caused by the fault. The reader will detect something wrong and will not consider the tag response. This is the service failure due to the previous error.

2.5.2 *Interaction faults that include all external faults RFID Heterogeneity and complexity:*

The main objective of this work is to develop an RFID emulation platform capable of validating both software and hardware solutions and aiming at increasing UHF RFID systems reliability. It must also take into account the problems between tags and effect of a faulty tag on the complete system. Due to the RFID system heterogeneity, this platform should be able to take into account all the system components and all the possible error sources: multiple tags or readers interactions, communication channel errors, disruptions of the environment, radiations, and threats of attack, etc...

As detailed in the previous section, RFID systems are composed of many tags, readers and a middleware. These components consist of analog, digital hardware and software parts. An RFID tag exchanges with a reader via an RF channel, but in the same time this channel can be shared with other tags as illustrated in Figure 2.24. So when validating a tag, one may first validate that the tag fulfills the specifications, and then when a robustness validation is engaged, it is also necessary to consider all system configurations. A faulty tag can impact other system elements but also other elements may disturb the tag under verification. Such a validation may then require to simultaneously consider all elements as explained later on.

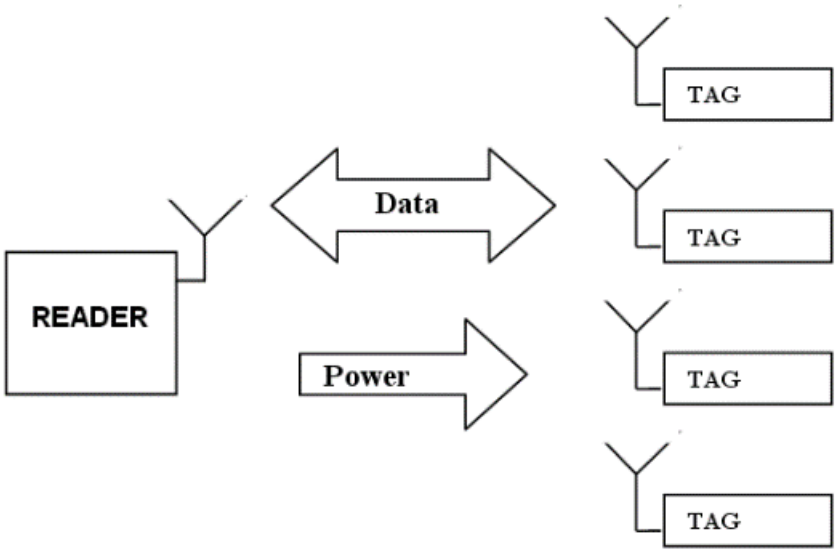


Figure 2.24: RFID System with several tags

RFID system has several abstraction levels as shown in Figure 2.25

- RFID application layer
- RFID software layer
- RFID hardware layer

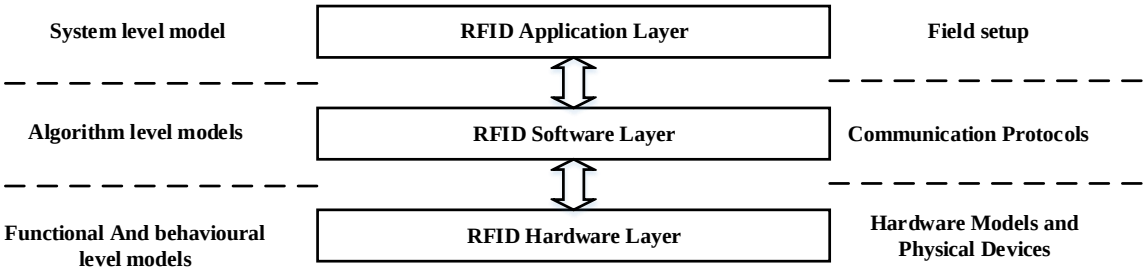


Figure 2.25: Layers for UHF RFID system modeling

Figure 2.25 shows the complexity of the model for each layer. At the top, the application layer defines the parameters of the hardware component layer that manages the functionality of the whole system. Software layer defines the algorithm to control communication links and data transfers between reader and tags, the communication protocols like collision arbitration, the possible authentication protocol. The hardware layer defines the physical structure of devices and the air interface [EPC2008]. Figure 2.26 shows the hardware layer component structure.

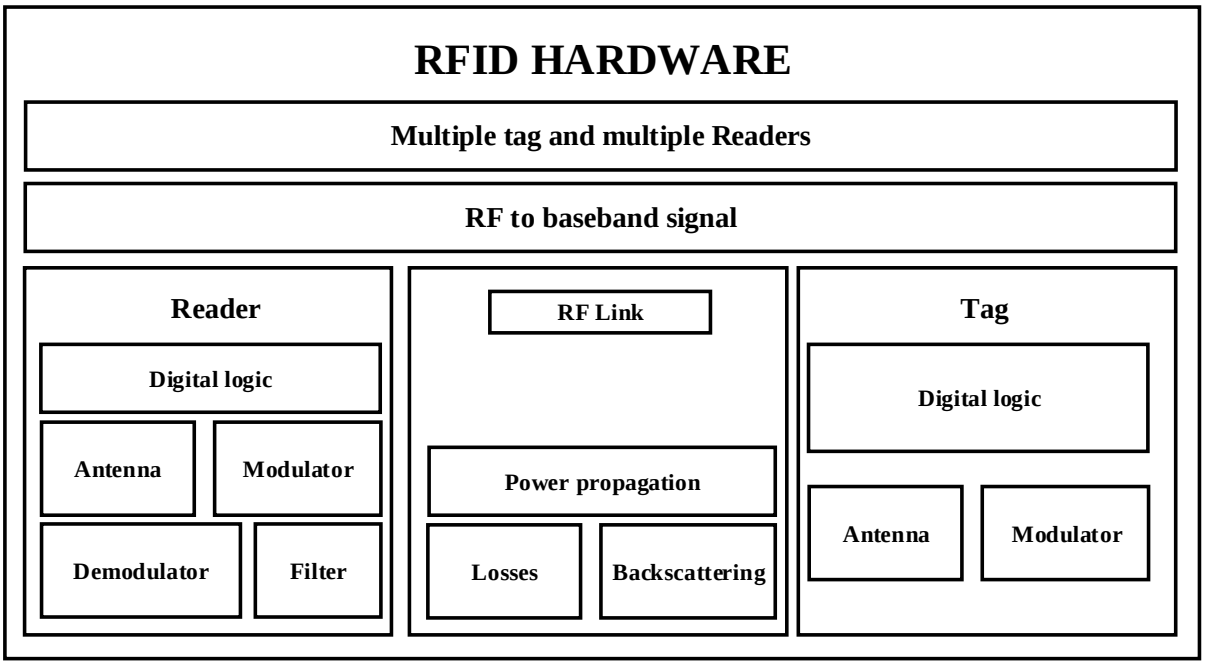


Figure 2.26: Hardware layer structure for UHF RFID system modeling

RFID system improvement needs to check two aspects: the first aspect is the requirement of the system level and the second concerns the hardware level. At system level, functionality and performances verification are required for the "End user" .At hardware level, the designs need to be verified for their conformance to regulations and protocols.

RFID simulation requires a large knowledge on the system and its environment. Simulation can be very time-consuming, in particular if it is necessary to test a lot of environment configurations. It is not possible to meet all RFID component requirements in one simulator. So one alternative to simulation is hardware emulation. First, emulation is much faster than simulation. Secondly, emulation can be used to observe the behavior of an entire system in realistic conditions. This justifies the use of an emulator to validate a complex and heterogeneous system as RFID.

2.6 Conclusion:

In this chapter, we have given a wide overview of RFID technology, by detailing some technical aspects of the structure of RFID system components. Different technical details on reader architecture, tag architecture, and the protocol used in this works [EPC2008] have been detailed in order to better see later implications on dependability issues. The second part of the chapter shows that the use of RFID technology in different critical application requires to study the robustness and the security of such a system. The heterogeneity of the RFID system composed of different technologies (RF, analog, digital hardware, and software) and numerous

components (several tags, reader) communication channels make the robustness enhancement a complex task. For these reasons, we propose an RFID emulator platform that allows us to carry out this task taking into account all the RFID system parameters.

Chapter 3

3 RFID validation platforms and robustness evaluation platforms

3.1 Introduction

In this chapter we will give a state of the art on different RFID system validation platforms and a brief description of existing robustness evaluation tools.

In the first part of this chapter, we will begin by describing the RFID system simulation based platforms and then we will give more detailed descriptions on RFID system emulation platforms.

In the second part of this chapter, we will discuss the methodology of fault injection**.** Then, we will finish this chapter by a state of art on different fault injection platforms that are proposed in the literature. We will distinguish between two categories of fault injection platforms: one based only on simulation and other based on emulation or prototyping.

3.2 RFID Simulation platforms

Usual techniques for validating RFID IC often use simulation because it offers a very good control and monitoring on internal signals. Several RFID simulators have already been developed [Land2012]. These simulators allow:

a) Simulating the communication protocol between the tags and the readers (called "air protocol")

b) Simulating the interactions between RFID readers and the middleware (server).

RFID Middleware designers generally use these simulators to perform a functional verification of their design. In [Palazzi2008], the authors present a case study using their RFID simulator called Rifidi. This tool simulates the reader/client interface of an RFID reader. But, neither the tag internal functioning, nor the communication between tags and readers can be simulated using Rifidi simulator. Indeed, the information about the tags in the reader field is directly applied on the reader model. Hence, Rifidi fits with RFID middleware deployment issues only. So, Rifidi is not useful for studying the dependability of RFID tags by fault injection. In [Angerer2009], the authors propose an RFID simulation and prototyping system. Although the system can handle reader-tag requests, it mainly focuses on the tag IC and its RF front-end. The digital part is described using the SystemC Library [SYSC]. The analog parts, i.e. modulation, demodulation and signal propagation, are modeled with Matlab, but the complete system co-simulation is performed in two separate steps: the digital part is first simulated and then the analog part is considered. So, it is not a real co-simulation environment with dynamic interactions between digital and analog parts. Moreover, this solution is very time

consuming and requires a high performance computer. In [Floerk2009] Floerkemeir et al. present the RFID simulator RFIDSim, which is a complete RFID simulator, nevertheless its main goal is to evaluate only RFID protocols.

In [Fritz2010-a] [Fritz2010-b], another RFID simulator called SERFID (Simulation and Evaluation of RFID Systems) has been developed to simulate RFID systems and has also been used to perform fault injections by altering the communication BER (Bit Error Ratio). In this work, due to simulation time constraints, no fault injection has been done in the tag IC. In this simulator all digital functions in readers and tags have been modeled using the Transaction Level Modeling with Distributed Time (TLM-DT). In order to model global and local environment effects, the RF links between tags and readers can be modified. This simulation is close to the RT level modeling. But, the evaluation of the robustness was only studied against channel communication noise that affects the tags and the readers during communication. In [Azambuja2008], the authors propose an RFID simulation considering the influence of some environment variables and physical characteristics of tags such as tag speed in relation to the reader, distances, and amount of tags that are going to be read simultaneously. In [Mirowski2009], the authors propose a simulation platform called Tyrell, the end-user can instantiate the values of a real systems and configure attacks into the simulation platform such as introducing clone tag attacks…etc.

Most existing RFID simulators take into account only a few parameters of entire RFID system parameters such as the parameters related to antenna in [Khouri2004], or the parameters of the RF circuit in [Yifeng2004], or they simulate only a part of the RFID system like [Mirowski2009] that simulates the RFID reader only by using java language. It appears that simulation in order to perform a complete RFID system validation appears unrealistic. In fact, evaluating the robustness of real applications running over RFID systems is very difficult using simulation because RFID simulator can only take into account a few parameters of the entire RFID system. Moreover the huge simulation time is another disadvantage of RFID simulation. As an example in [Abdelmalek2013] the simulation duration for the data transfer between a reader and a tag during 200 seconds takes 3 days using a workstation. However using emulation it will take only 200 seconds. This justifies the use of emulation. Table 3.1 summarizes some of the qualitative differences between the simulation and emulation in realistic environments.

Table 3.1: Experimental platform comparison

	Simulation	**Real world**	**Emulation**
Done in real time	No	Yes	Usually
Easy to control	Yes	No	Yes
Cost of use	Low	High	Medium

3.3 RFID Emulation platforms

To our knowledge, there is no complete prototyping platform for UHF RFID system development. Several prototypes have been developed, but they only focus on one aspect of the entire system, i.e. the reader or the tag. In this section, we present the prototyping and development platforms existing in the literature.

Several RFID reader prototypes have been developed. In [Angerer2008] the authors describe the implementation of a dual frequency RFID system. This platform is designed to operate in the 13.56 MHz and 868MHz ranges, and is based on FPGA and DSP to implement protocol stack and analog parts that function at the HF and UHF ranges. The codes for the DSP processor and the FPGA are generated by MATLAB and Xilinx System Generator and this code generation is not optimized.

Roy et al. in [Roy2006] describe the architecture of an UHF RFID reader based on FPGA. The FPGA architecture is described as well as potential interfaces that can be used.

SDR (Software Defined Radio) have been used to develop a customizable reader in [Buettner2012]. In such a SDR system, all the DSP functionalities are done through the host PC, while the acquisition, ADC and DAC are realized through an external device, the Universal Software Radio Peripheral (USRP). This system suffers from the narrow bandwidth available in the USRP, as well as the timing delays introduced in all the processing done on the host side.

Several works propose several tag prototypes. The Wireless Identification Sensing Platform (WISP) [Sample2008], designed by the Intel Research group [Buettner2004] is one example of an open source UHF RFID tag development platform. The platform proposes a passive tag, consisting in an MSP430 microcontroller with sensors attached to it. The platform has the disadvantage of being very low-range (limited to less than 3 meters) as well as not being fully EPC C1 Gen2 compliant, i.e. it does not implement all required EPC commands. Its firmware cannot handle complicated signal processing or complex algorithms. Finally this platform is only a firmware implemented in a microcontroller, and can’t reflect the RFID UHF tag hardware design. It can mainly be used for protocol validation.

Another open source development platform for prototyping UHF RFID tags is the semi-passive development tag platform based on the PIC24F microcontroller proposed in [Li2012]. The semi-passive nature allows this tag to have a better range than WISP. This tag has been designed to have an extension support for sensors and an easily modifiable code for researchers to experiment their solution on RFID Systems. A similar system implemented on an FPGA is presented in [Chen2011]. This work only focuses on rapid UHF RFID tag simulation it uses an integrated Analog front, implemented on 0.18µm one-poly six-metal CMOS process, but this platform is not suitable fault injection.

There are also RFID protocol analyzers that can be used for debugging the air interface, analysing performances, and adjusting parameters. In the field of UHF RFID, Donno et al. [Donno2011] propose an RFID receiver system, based on GNU Radio and implemented in a USRP. The receiver contains a filter and a channel tracking implemented in Software Defined Radio. This prototype has been developed to RFID protocols and RFID based localization.

Further researches were conducted in [Catarinucci2011. In this work the platform is used to evaluate the performance of a UHF RFID system. But the use of the USRP makes this platform expensive. A similar platform is implemented in [Athalye2011]. A system with additional transmitters was proposed and implemented in [Jun2010]. In this work, a continuous wave transmitter is used to extend the forward link range of the UHF RFID system. The forward link (from reader to tag) is the weakest link in an RFID system [Dobkin2007] if the tag does not have enough power to be power on. The continuous wave transmitter addresses this issue. This solution can effectively increase the range of passive tags.

A security evaluation for UHF RFID is implemented in [Naray2010] by designing a blocking reader to prevent any unauthorized reading of RFID tags. The design is based on the TI CC1101 chip transceiver and a microcontroller.

The programmable IAIK RFID Demo tag [iaik2014] can be used to emulate real RFID tags. It consists of an antenna, an analogue front-end and a programmable Atmel ATMega128 microcontroller that supports the EPC C1 Gen2 protocol [Hutter2008]. As it is semi-passive, IAIK RFID Demo tag has an onboard power source to power the microcontroller. This semi-passive tag emulator is mainly designed for security functionality as shown in the work [Feldhofer2010] that uses it to analyze the security against side-channel and fault attacks. The details of the design of this tag, schematic or layout are not provided. As they did not reveal their analog or digital parts, we cannot assess or modify their design to meet the requirements of our research project. The price of IAIK RFID Demo Tag is about 650$ to 750 $ per tag,

which is very expensive to be used in the case of UHF RFID system emulation with numerous tags.

Table 3.2 presents a comparison amongst different RFID system emulators existing in the literature. We note that no emulator using FPGA in the literature implements all the commands of the standard protocol EPC C1 Gen2.

Table 3.2: Survey of RFID emulator

Paper	**Functionality**	**Device application**	**Hardware / software**
Angerer[Angerer2009]	Dual-frequency tag Prototyping (HF and UHF)	Prototyping	FPGA Virtex II and TSM320s DSP
Roy[Roy2006]	UHF RFID reader	Prototyping	FPGA Virtex-4
Buettner[Buettner2012]	UHF RFID reader	Prototyping	GNU Radio and USRP
Sample[Sample2008]	Semi-Passive tag platform	Prototyping	TI MSP430 microcontroller
T. Li [Li2012]	Semi-passive tag platform	Prototyping	PIC24F microcontroller
Feldhofer[Feldhofer2000]	Semi-passive tag Platform	Security	Xilinx FPGA
L. Chen [Chen2011]	Semi-passive tag platform	Tag simulation	Altera FPGA
Donno[Donno2011]	Receiver	Performance analysis, Localization	GNU Radio and USRP
Park[Jun2010]	Transmitter	Forward link extension	CC1110 transceiver and microcontroller
Athalye[Athalye2011]	tag signal interceptor	Localization	Custom UHF RFID tag on FPGA
Kenarangui [Kenarangui2010]	RFID reader with camera and image processing	Localization	RFID and image processing software
Narayanaswamy [Naray2010]	Blocking reader	Security	CC1101
Demodtag IAIK TU Graz [iaik2014]	Semi-Passive tag platform	Security	ATMega128

As seen in the two previous sections, a lot of RFID validation platforms are proposed for the simulation or the emulation of a part of a RFID system. Each platform or prototype analyzes only one aspect of all the possible aspects existing in RFID system. In addition, it remains difficult to predict the behavior of a RFID system in harsh environments where there are intentional or environmental disturbances. These disturbances can involve the deterioration of a part or several parts of the RFID system. There is a need for a platform that can emulate this type of disturbance. In the next section, we will present the principles and the architecture which allow us to emulate such faults into a RFID system.

3.4 Fault injection

The fault injection in an UHF RFID tag digital IC is a complex task. This task necessitates (1) determining the **types of faults** to inject into the IC, (2) **planning the fault injection process,** (3) **injecting** these faults, and (4) extracting the data to analyze their effects. This section explains the different types of fault injection. To realize a fault injection campaign, fault injection model and methodology to perform the physical fault injection are needed.

FARM model is the most known fault model and methodology [Arlat1990]. Our work, which aims at realizing a platform to analyze RFID tag fault tolerance, exactly follows this model. As far as we know, there is no existing work about RFID IC robustness evaluation, although the widespread use and deployment of this technology in our life and in many critical systems. In this work, we will use the FARM model to study the dependability of the RFID tag digital IC.

3.4.1 *Farm Model*

FARM fault injection model has been developed in LAAS-CNRS (French) in 1990. This model describes a methodology for system dependability evaluation using fault injection as shown in Figure 3.1.

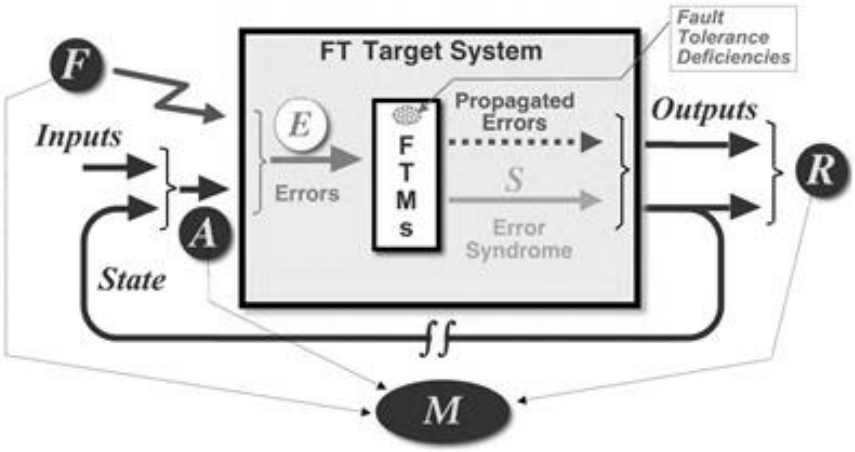


Figure 3.1: FARM model [Arlat1992].

As shown in Figure 3.1, this model contains 4 sets [Arlat1992] F, A, R, and M. F is the set of all anticipated or perceived faults to which the target system could be exposed over its operational phase. A is the set of the inputs of the system aimed at exercising the injected faults. R is all data and information collected during the fault injection experiment. It helps knowing the behavior of the system in presence of faults. In the FARM model, experimental results are the estimated coverage of detected faults, the silent faults, the fault dictionary entries, etc. These

results are defined by the set M. The FARM model shown in figure 3.2 is a modified version of the previous original model. This new model is better adapted for digital IC fault injection [Elks2005].

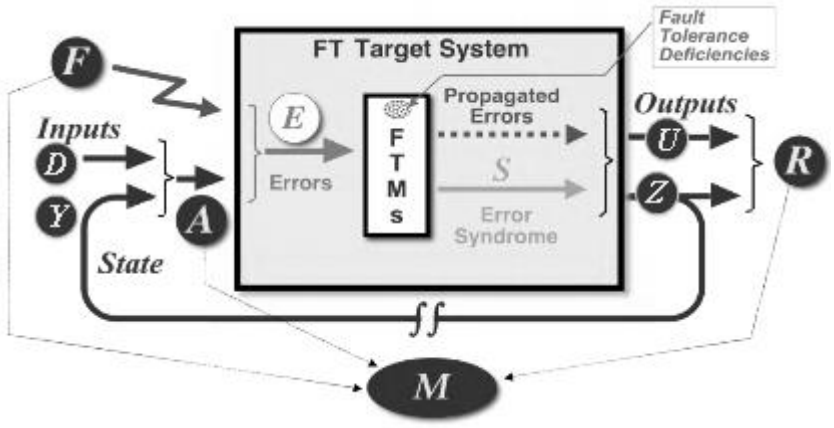


Figure 3.2: Modified FARM model [Elks 2005]

In the Figure 3.2, the set Z defines the internal state of the digital IC. This attribute expresses the idea that a digital system is based on internal states. Digital IC behavior is discrete time-based with transitions from one state to a set of other states. The Y set defines the current state, input event change, etc. The set D defines the external input data. The set U are the outputs of the circuit.

3.4.2 *Fault Injection Techniques*

As shown in figure 3.3, fault injection is classified into three categories:

1) Physical-based fault injection
2) Simulation-based fault injection
3) Emulation-based fault injection

In physical-based fault injection, the target IC is the same digital IC that will be deployed in the field of operation. In simulation-based fault injection, the fault injection will be done in a model which describes the behavior the digital IC. For example, for RFID robustness evaluation, simulation-based fault injection has been used in [Gilles2011] using SystemC models. Emulation-based fault injection uses hardware prototypes, and in this case the injection is done directly in the designed IC architecture.

In this work, we will evaluate the RFID digital IC dependability by studying the effects of *Single Event Upset error (SEU).* This fault is the most experienced by modern IC.

A SEU is an error that occurs in the logic circuit and causes a bit flipping to a wrong logical value. All fault injection approaches that we will discuss are shown in Figure 3.3.

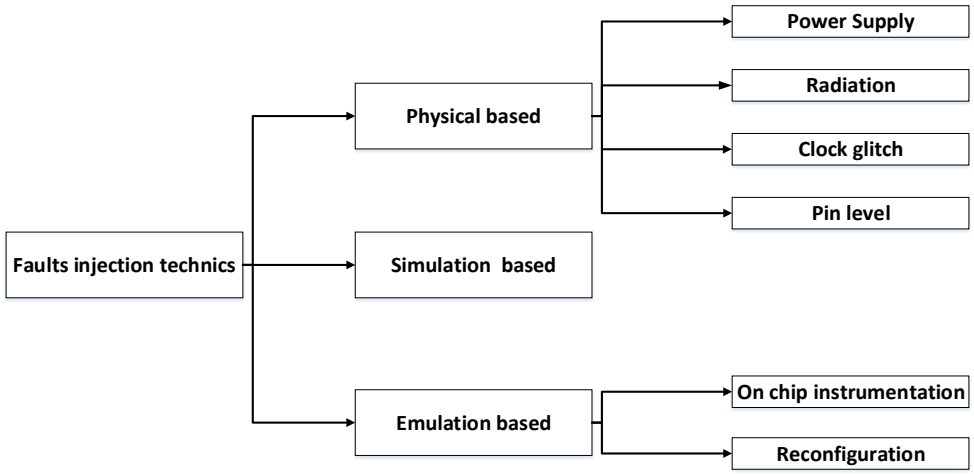


Figure 3.3: Taxonomy of fault injection techniques

3.4.2.1 *Physical-based fault injection*

a) Pin level fault injection:

Numerous works have been developed in the 90s for pin-level fault injections. This technique uses pins to introduce faults in digital circuit. Several pin-level fault injection tools have been developed (e.g., MESSALINE [Arlat1990 (a)] and RIFLE [Madeira1994]). The advantage of these tools is that they are not invasive, but hardware fault injection can damage the digital IC. High integration density of mixed signals and dense packaging technologies limit accessibility to pin-level fault injections. The major problem of pin-level fault injection is the model definition of internal faults inside the circuit.

b) Power Supply Fault Injection:

In [KARL1991] fault injection approach via power supply disturbances has been proposed. This approach proposes to inject faults by modifying the supply voltage Vcc of digital gates or IC. This approach leads to transient faults by the voltage Vcc drop. The disadvantage of this approach is that it requires special circuits to disturb the power supplies of IC.

c) Radiation Based Fault Injection

Radiation fault injection can be done using several methods as Electro Magnetic Interference (EMI), Laser Fault Injection (LFI), etc. [KARL1995] shows how the injected fault with an EMI can propagate inside the circuit. In one other hand, laser allows injecting faults

known as soft-error. LFI has a good controllability in both space and time. In addition, LFI is not invasive. There are several works on LFI as [Sampson97], [Duzellier-00], [LEWIS-05]. This fault injection technique is characterized by its spatial accuracy, the spot size and frequency, but LFI requires a sophisticated and expensive material.

d) Clock Glitch

Glitches in the external clock signal generate faults. There are several works about this type especially on cryptographic circuits [Agoyan2010] [Fukunaga2009] [Selmane2008]. In [Hutter2008] glitch and optical inductions attacks in RFID chips have been performed. The authors provide concrete results of practical physical-based fault injection on RFID devices. They use local and global fault-injection methods directly on RFID tag digital IC. The global injection impacts all the chip, and the local fault injection only impact a specific area of a RFID chip. For example, the local fault injection prevents writing data into the tag memory or involve writing faulty values. The main intention of this work is identification of potential weaknesses of RFID digital IC. The authors conclude that RFID tags do not include any countermeasures against these types of fault.

3.4.2.2 *Simulation-based fault injection*

Simulation-based fault injection can be performed at different abstraction levels, such as the gate level, RTL level, and system level.

The principle of the simulation-based fault injection is to compare the simulation results without and with fault injection by using simulation tools. FOCUS **[Choi1992]** is an example of a simulator for fault sensitivity analysis of designs with respect to error.

In simulation-based fault injection, the circuits are described using a Hardware Description Language (HDL) such as Verilog or VHDL. In this context we find in the literature several works that automated the fault injection experiments with HDL models, as the MEFISTO tool [Jenn1994] and [Delong1996].

In simulation-based fault injection, the fault injection is based on the modification of the hardware description [Assaf2004] by modifying the HDL description through scripts to simulate for example stuck-at faults. The advantages of simulation-based fault injection are the high controllability, observability, repeatability and reproducibility. As simulation is done in a controlled environment, it supports a larger set of fault models than physical fault injection. But the disadvantages of this fault injection method are the accuracy of the system model, the fault model takes a lot of time to be developed, and the fault injection by simulation is very time

consuming. In [Fritz2012] the authors propose a RFID fault injection simulator called SERFID. SERFID allows injecting faults into the tags, the readers or the communication channels to simulate the effects of different failures. These failures may appear due to the low quality of the communication, low tag powering, errors in the tag memory, etc.

3.4.2.3 *Emulation-based fault injection*

The FPGAs provide new opportunities for emulation or hardware-based fault injection. In fact, they provide high flexibility for the prototype development.

Simulation-based fault injection requires a huge time and a powerful computer, which makes the simulation of complex circuits infeasible. To speed up this process the HDL designs can be evaluated by hardware emulation on FPGA. The first published work in this area was proposed in [Kwang-Ting 1999]. In this work a method for fault emulation is used for assessing the fault coverage of tested circuits.

Based on the literature we distinguish two environments for the FPGA-based fault injection. The first one is based on the circuit **instrumentation** and the second one is based on the manipulation of the FPGA configuration file, which is called **Reconfiguration-Based Fault-Injection.**

a) Reconfiguration-based fault injection

In SRAM FPGAs, the configuration file contains data and logic configuration. The configuration RAM cells can be considered as a shift register. The configuration data is serially downloaded from the host computer to the FPGA. This data and the content of all the memory cells can then be read back from the FPGA. Once an FPGA has been configured there are two methods to reconfigure it. The first method is static and is called Compile-Time Reconfiguration. The second method is dynamic and is called Run-Time Reconfiguration. In the case of a Compile-Time Reconfiguration, to implement any changes it is necessary to recompile and re-synthesize the modified blocks in the system, then regenerate the configuration file for the complete system, and download this new configuration on FPGA. Run-Time Reconfiguration provides the ability to configure the FPGA during the execution of an application. It is possible to reconfigure the entire circuit or only a specific part of it, this is called partial reconfiguration. In a partial reconfiguration, the un-reconfigured blocks can continue to operate normally. The fault injection by dynamic reconfiguration is done directly by modifying the configuration data.

The main objective of dynamic reconfiguration is to save time consumed by the different steps (compilation, synthesis, generation of the full configuration file) of the static reconfiguration. The experiments presented in [Antoni2001] have been performed using two different FPGAs, one of them have partial reconfiguration mechanisms. The bitstreams (configuration files) are modified with JBits, a Java tool that allows a reading of the FPGA configuration and quick and easy reconfiguration. The results obtained demonstrate the feasibility of the fault injection by reconfiguration and show that it allows significant savings of time under certain conditions. The University of Seville and the European Space Agency have jointly developed the FT-UNSHADES system [Tombs2004]. It uses two copies of the circuit to be analyzed, one used as reference and the second used for fault injection. The state of the two copies allow detecting the activation or not of the injected fault. The high speed connection between the host computer and the FPGA is one of the advantages of the presented environment. The use of two copies accelerates the comparison of the results, and also reduces significantly the memory space required in the FPGA.

To reduce the reconfiguration time induced by the partial reconfiguration, [Kafka2006] uses a FPGA embedded processor to perform the reconfigurations. The data required to select the fault injection target (LUTs, multiplexers ...) are recovered using placement & route tool. All necessary configurations are prepared before the fault injection campaign to be uploaded. During the campaign there is no data transfer between the host PC and the FPGA. The results of classification errors are generated by a hardware comparator and returned by the FPGA to the host PC at the end of the campaign. This work not only shows that the approach is feasible but also that it is faster than a simulation-based approach. However, it can require a lot of memory space if a large number of predefined configurations need to be stored. Moreover, as for the previous approach, the implementation of the two copies of the analyzed circuit (golden and faulty) halved the size for this one.

b) Instrumentation-based fault injection

This method is based on the modification of the original description in order to allow the injection of faults. The circuit is first instrumented, then synthesized and finally emulated. First prototyping for the safety analysis has been done at TIMA laboratory in 1999 [Leveugle1999].

The instrumentation and the generation of behavioral models to represent error propagation modes are also presented as effective techniques for safety analysis. We will give more details in chapter 4.

FPGA-based fault injection approach has been developed by Politecnico of Torino as presented in [CIVE-01a]. It allows the injection of bit-flips in the memory elements of the instrumented circuit. This tool consists in two software modules, and a mixed module (software / hardware). The software modules generate the fault list from the circuit netlist, apply the input vectors and analyze the results. The mixed module includes the instrumentation tool, the injection control tool and the hardware interface with the FPGA [CIVE-01a]. As shown in Figure 3.4, the system is composed of three modules:

1) **Fault List Manager:** The role of this module is to analyze the circuit in order to generate a fault list. This list manages the time and the place where the fault will be injected in the circuit.
2) **FI Manager:** The role of this module is to manage the fault injection.
3) **Result Analyzer:** The role of this module is to analyze the data produced by the previous modules, to classify the faults according to their effect, and to make statistics on each type of faults.

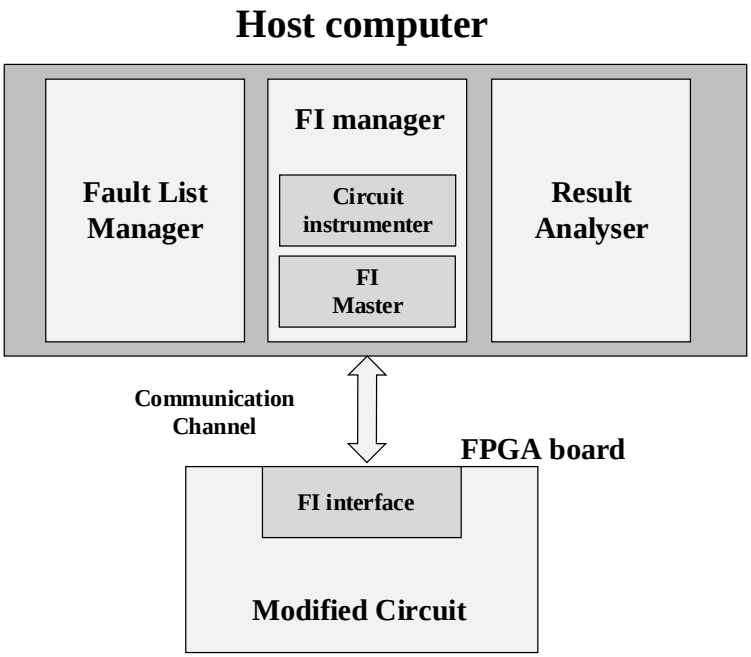


Figure 3.4: Instrumentation based register mask [Civera 2001B]

The FI Manager is implemented both in the host PC and in the FPGA (with the emulated circuit). This module is composed of 3 parts:

- **Circuit Instrumenter:** it is a software necessary to modify the circuit to support the Fault Injection (FI).
- **FI master:** it uses the fault list to select the time and the place of the fault injection.
- **FI Interface:** this module interprets and executes the commands of the FI master module.

Another developed FPGA environment has been developed by the Microelectronics group at the University of Madrid. It can implement two possible modes: the first using connection of memory elements called state-scan and a second with a mask register called mask-scan [Portela2004]. In the state-scan mode, the scan line (or shift register) is used to download the faulty states into the memory elements of the circuit. Instrumentation adds very few logic as one multiplexer for every latch is required. Faulty states serially loaded must be defined before the beginning of fault injection campaign and are stored in RAM. In mask-scan mode, a mask is applied to the memory elements of the circuit for inverting the contents of those targeted elements. The latter technique is taken from [Civera2001a]. For the two modes, the circuit analyzer, the fault injection controller and the result analyzer are emulated by the programmable logic, and the RAM banks are used to store data. The software part allows the generation of the instrumented circuit, the generation of the fault list, the access to the memory banks, and FPGA

configuration. The instrumentation is done at netlist level after synthesis. The scan-chain approach is the most time consuming.

Three injection modes are proposed in [López2005]: the state-scan mode, the mask-scan mode and a new mode called temporally time-multiplexed. The novelty is that most tasks (control of fault injection, errors classification) are performed by hardware blocks to limit the transfers between the prototype and the host computer that are very time consuming. The mask-scan mode implements the same instrumentation and the same characteristics (shift register ...) as proposed in [Civera2001a] and described above. The only difference lies in the injection controller that is implemented in hardware.

3.5 Conclusion

In this chapter we have discussed about the different methods to perform early analysis in the RFID system design flow. We have presented several RFID simulators with their advantages and limitations. Then we have discussed about the RFID emulation platforms. Finally, a table comparing the advantages of each platform has been given. In the second part of this chapter, we have discussed about the robustness analysis of RFID system. This analysis is performed thanks to the faults effect analysis on this system. We have discussed about the different types of fault injection. Our work in the next chapter will be based on the instrumentation based fault injection method.

The purpose of our work is to develop a design method based on a hardware emulation platform dedicated to RFID tag dependability and security study. Emulation permits to evaluate the UHF RFID tag within its real environment. Using this approach allows us to take into account all the parameters of the RFID system. This emulator based approach allows a better understanding of the behavior of the tag in a realistic environment, and allows rapidly deploying research on operational RFID system.

An RFID emulator can be used within an RFID environment including reader and other tags or even other emulators as depicted in Figure 3.5

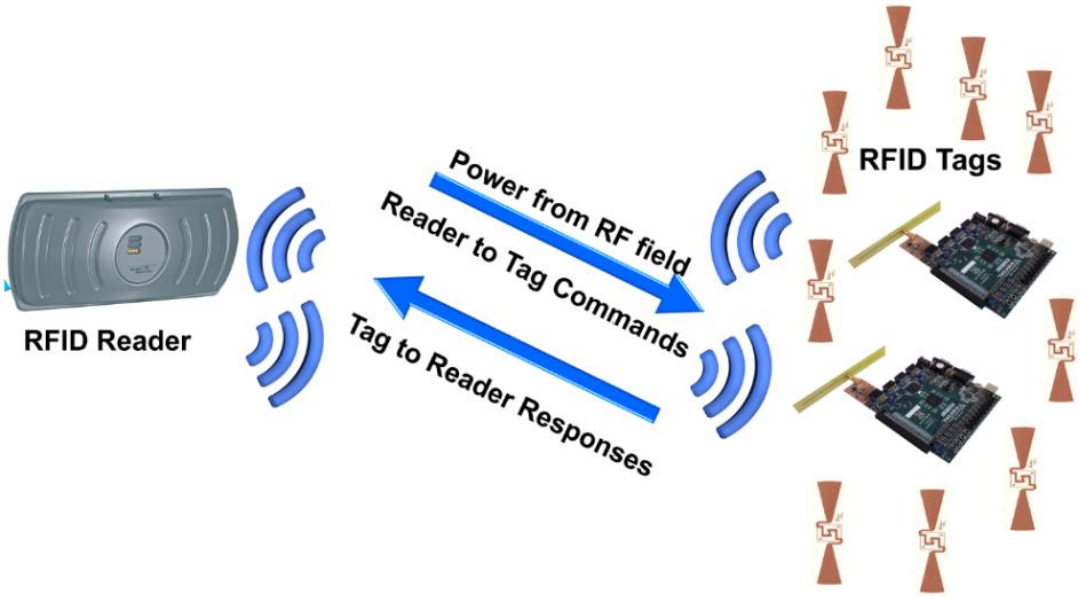


Figure 3.5 RFID tag emulation environment

Emulation of RFID system within a real environment allows:

•Measuring the performances of the digital architecture under validation (for example, the Read Error Rate RER, ratio between the number of successful read and the number of read error [Fritz2012]).

•Observing internal state of the tag in order to monitor its behavior.

•Injecting faults by changing functional or structural parameters to respectively model fault attacks and environmental disturbance effects (like bit flipping).

•Controlling internal state of the tag to perform efficient verification

Chapter 4

4 RFID emulator and robustness evaluation platform

4.1 Introduction

As we have seen in the previous chapters UHF RFID technology is used in many critical applications that require a lot of caution to ensure the working of systems in any environment. Thus, designing RFID IC dedicated to secure or robust applications raises critical issues. It is particularly difficult to validate early in the design flow that the circuit will be robust against security attacks and environmental perturbations, and also to measure the effects of the countermeasures on the system performances. Thus, in this chapter we will present an RFID tag emulator with fault injection capabilities to emulate errors in order to validate the robustness of both the tag architectures and the protocol. This emulator also allows internal monitoring of the tag in order to analyze faulty behavior.

Due to their heterogeneity, RFID systems are complex to validate and analyze. Thus the verification of a whole RFID system using computer model simulations is very time consuming and can hardly consider all the system parameters.

The proposed emulator should help to evaluate faulty tags influence on the whole system by injecting faults in RTL HDL descriptions and thus to analyze very early in the design process the consequences of faults on the behavior of complex RFID systems.

In the first part of this chapter we will present our EPC C1 Gen2 tag emulator. We will explain in more details the most important components of an RFID tag, especially the ones that have a direct effect on the performances of the tag. In the second part, we will present the fault injection capabilities of the developed platform. In the third part, we will present the experimental results of the fault injection that have been obtained with the platform. Finally we will show the experimental results of two kinds of countermeasure to enhance RFID tag performances against SEU Faults.

4.2 Proposed Emulator Architecture

4.2.1 *Description of the emulator*

In this section we will discuss how the EPC C1 Gen2 protocol has been implemented in the FPGA to emulate the digital part of the RFID tag. We will also describe the analog front–end used to demodulate the signals coming from the reader, and to modulate the responses of the tag to the reader. Figure 4.1 shows a high level architecture of the EPC C1 Gen2 tag design.

This architecture is made of two main parts: the digital one (UHF digital baseband) and the analog one (AFE for Analog Front-End), which are both described in details in the following.

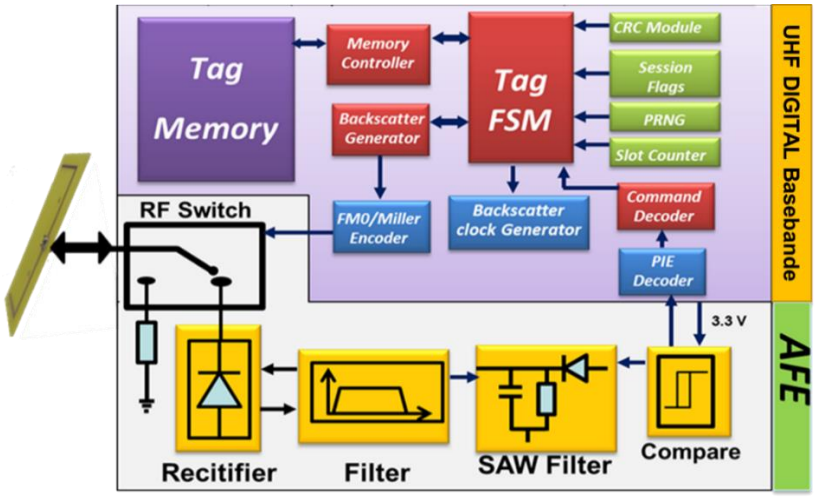


Figure 4.1: EPC C1 Gen2 tag block diagram

4.2.1.1 *UHF RFID tag Analog front-end*

Analog front-end is one of the key elements in a passive RFID tag. It is responsible of numerous operations: RF signal to DC conversion, voltage regulation, voltage limitation, and modulation and demodulation of the incident RF wave to deliver a digital baseband signal to the digital part of the RFID tag. Figure 4.2 shows the Analog Front-End (AFE) that we have used in our emulator. It consists in a rectifier, a band-pass filter, a Surface Acoustic Wave (SAW) filter, and a comparator.

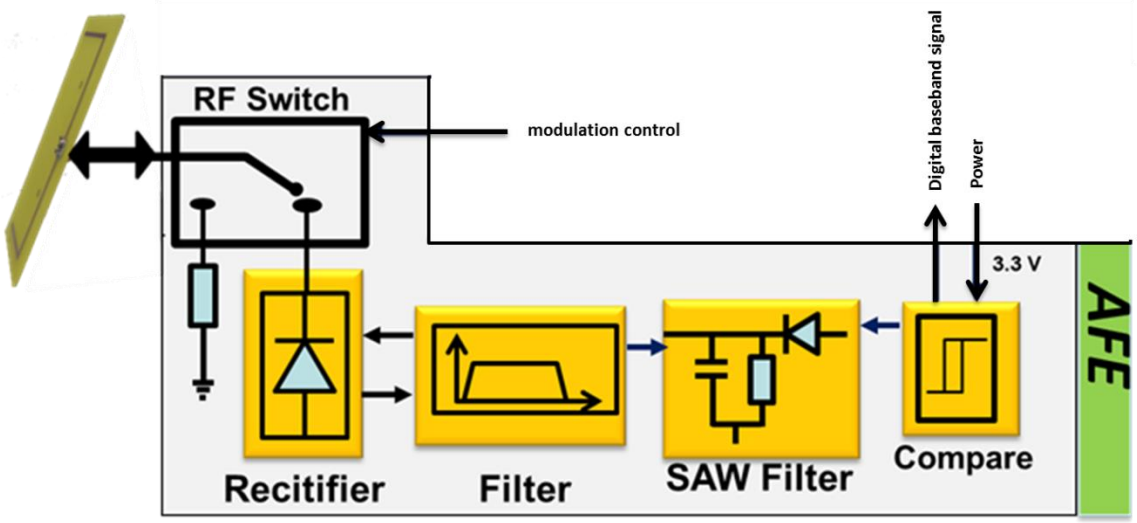


Figure 4.2: Analog Front-End architecture

The ASK signal consists of bursts of sinusoidal signal generated by the RFID reader. When it passes through the rectifier we obtain the positive half wave signal. The resulting signal passes through a low-pass filter and SAW filter to obtain an envelope detection. The band-pass filter and the SAW filter output a signal with a better Signal to Noise Ratio (SNR). The comparator compares the value of the input signal coming from envelope detector (SAW filter) to a reference voltage. If the input signal voltage is larger than the reference voltage, the comparator outputs a high logic level, if the input signal voltage is smaller than the reference voltage; the comparator outputs a zero logic level. Thanks to this process the original binary sequence (baseband signal) is generated. The RF switch is used to control the backscattering modulation. In fact, the tag response is made with the reflection of the incoming waves coming from the reader. The reflection coefficient of the tag can be changed by turning on or off an RF switch as shown in Figure 4.2. If the replied data is "1", the switch turns on and the impendence of the tag is lowered. On the other hand, if the replied data is "0", the switch turns off and the impendence of tag is matched to Z (open impedance).

4.2.1.2 *UHF RFID tag digital baseband architecture (RT level schematics)*

In this subsection, we give a detailed description of the designed RFID digital baseband, which is used to handle communications and control operations of the RFID tag. This baseband is compliant with the EPC C1 Gen2 UHF RFID protocol [EPC2008]. The validation of this baseband will be done by doing several experimentations. Then, the robustness of this digital baseband will be evaluated and improved.

The main activities of this digital baseband are to decode the incoming data from the reader, to modify the memory contents and to execute the relevant commands to response to different reader interrogations. Figure 4.3 shows the proposed architecture of the designed digital baseband tag. It is made of 14 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, timer T1, timer T2, backscattering clock generator (BLF) and FM0 Miller encoder. In the following the purpose and the architecture of these modules will be described.

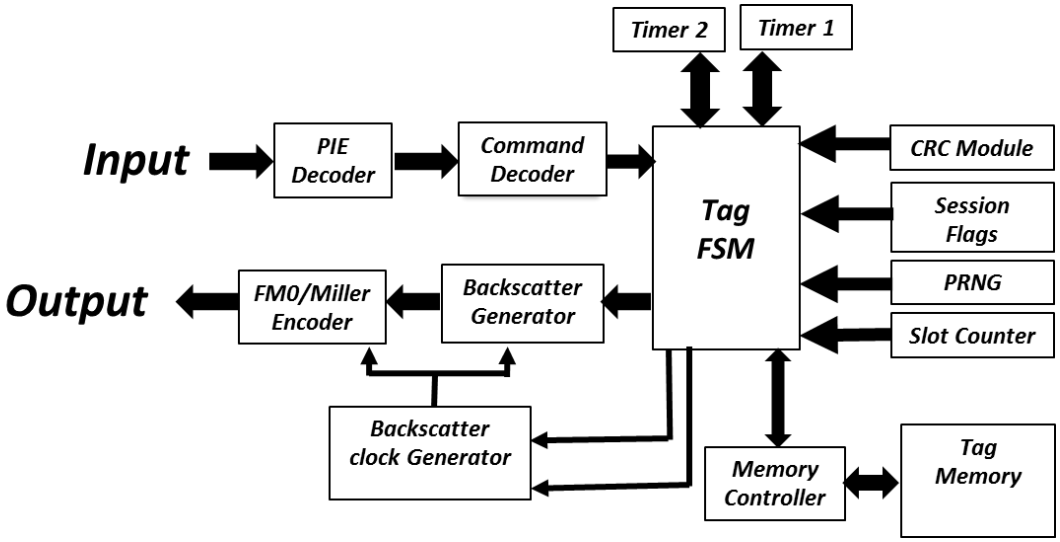


Figure 4.3: RFID Baseband architecture

4.2.1.2.1 PIE decoder

As explained in the chapter 2 (section 2.4.1.3)**,** the reader uses the Pulse Interval Encoding scheme (PIE) to transmit the data to the tag [EPC2008]. After signal is demodulated in the AFE, the PIE Decoder detects the level changes within the received signal by using an edge detector. Then a counter measures the duration of each level as shown in Figure 4.4. All measured times are stored in a register, and are then evaluated in order to define the main communication parameters described in the standard. The most important parameters are the Tari, TRcal, RTcal (shown in Figure 2.12 and Figure 2.13), and the Pivot that are used as time references to decode data0 and data1 is defined as:

$$Pivot = \frac{RTcal}{2} \quad \textbf{Equation4.1}$$

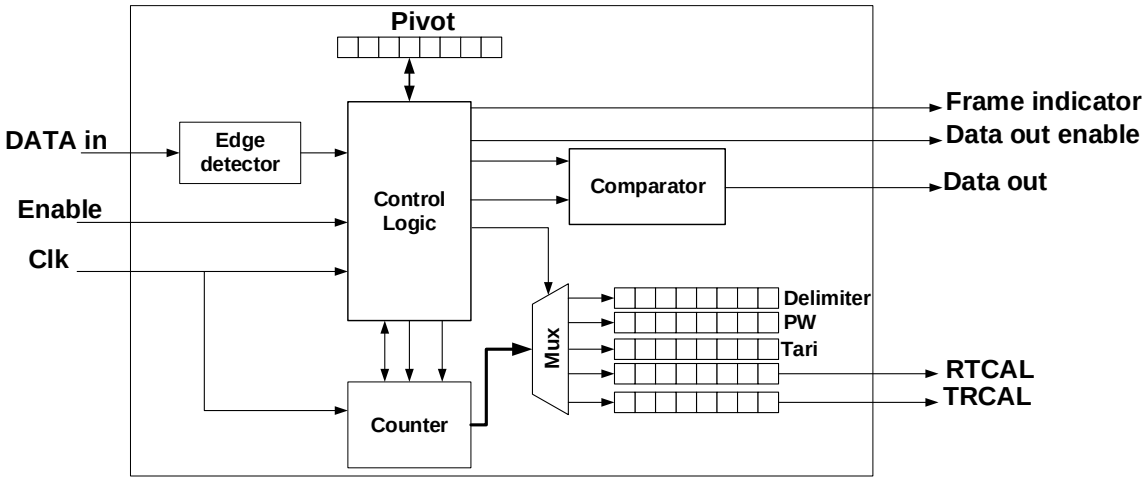


Figure 4.4: PIE decoder RTL schematic

4.2.1.2.2 Command decoder

As specified in EPC C1 Gen2 standard [EPC2008] the reader interrogates the tag by sending commands. Before discussing the command decoder module, we first give an example of command. We choose the Query command which has been previously explained in chapter2.

As shown in Figure 4.5, this command contains eight parameters which determine reader to tag link communication characteristics. The most important parameters are DR, M, and TRext. The DR parameter is used by the tag to compute the data rate of the tag-to-reader link as described in chapter 2 (Section 2.2.5.2). The M parameter is used to select the FM0 or one of the three Miller encoding schemes to encode the data backscattered by the tag. The TRext parameter determines if the tag frame response contains a preamble or not.

	Command	DR	M	TRext	Sel	Session	Target	Q	CRC
# of bits	4	1	2	1	2	2	1	4	5
description	1000	0: DR=8 1: DR=64/3	00: M=1 01: M=2 10: M=4 11: M=8	0: No pilot tone 1: Use pilot tone	00: All 01: All 10: ~**SL** 11: **SL**	00: S0 01: S1 10: S2 11: S3	0: *A* 1: *B*	0–15	CRC-5

Figure 4.5: Query command parameters

The command decoder is in charge to decode all the EPC commands by looking to the first 2, 4 or 8 received bits from the MSB to the LSB. After decoding the command name the length of this command (including all the parameters relative to this command) will be known as described in [EPC2008]. Figure 4.6 shows a part of the pseudo-code for the command decoding.

```
Begin
Detect the first decoded symbol
   If (1st symbol =0) then
Detect the second decoded symbol
If (2nd symbol =0) then
    Command is Query rep
 Else
     Command is Acknowledge
End
If (1st symbol =1) then
      If (2nd symbol =0) then
     If (3rd symbol =0) then
                ……
End.
```

Figure 4.6: Part of the pseudo-code of the command decoder

After detecting the command type and all the parameters included in its frame, the frame is checked using the CRC engine.

4.2.1.2.3 CRC5 and CRC16

Each frame includes a 16 bit Cyclic Redundancy Check (CRC) field for error detection. In Figure 4.5 the frame contains a field CRC5. The tag uses it to verify the integrity of the command frame. The EPC C1 Gen2 protocol [EPC2008] specifies the uses of two types of CRC: CRC5 and CRC16.

The developed tag contains both CRC5 and CRC16 encoders. Figure 4.7 shows the schematic of CRC5 encoder. The CRC5 encoder is based on a polynomial and a preset defined in the table 4.1. The CRC5 value is computed in several steps. Firstly, the CRC5 register is loaded with the value 01001_2 (preset value). Second, the data is pushed bit per bit toward the CRC input. At the end the register Q [4:0] holds the CRC5 value [EPC2008]. To check the CRC5 of the received data, the CRC5 will be loaded by the value 01001_2 and then the data and the CRC will be passed bit per bit (at every clock tick), at the end if we find the value 00000_2 (residue), this mean that received data is valid without error [EPC2008].

Table 4.1: EPC C1 Gen2 command

polynomial	preset	Residue
X^5+X^3+1	01001_2	00000

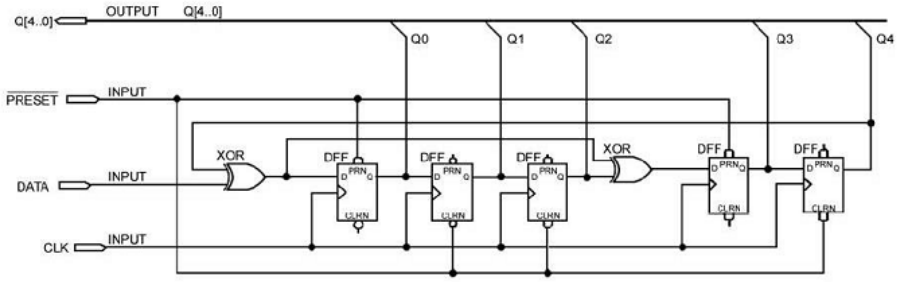


Figure 4.7: CRC5 circuit

4.2.1.2.4 FM0 and Miller encoders

Tag to reader link data encoding is done using either FM0 or Miller code as described in subsection in chapter 2 (Section 2.4.1.3). Miller and FM0 data encoding forms are based on M and TRext parameters which are extracted from the Query command frame at the beginning of every inventory round. Miller and FM0 data frame contains a preamble and an end of frame to

indicate the beginning and the end of each sent frame. Figure 4.8 shows the pseudo code of FM0 Miller frame encoding.

```
Begin
Get and decode query command
  Get M value from Query command frame.
  Get TRext value from Query command frame
Case based on M value:
       Case 0: M=0 use FM0 encoding
       Case 1: M=0 use Miler encoding with subcarrier =2.
       Case 2: M=0 use Miler encoding with subcarrier =4.
       Case 3: M=0 use Miler encoding with subcarrier =8.
   End case.
Case based on TRext value:
       Case 0: TRext = 1 preamble include violation of
encoding
       Case 1: TRext = 0 preamble without violation of
encoding
End case;
```

Figure 4.8: Pseudo code for tag to reader link frame encoding

Figure 4.9 shows the RTL schematic of the FM0 and Miller encoder block. At the beginning of every inventory round the tag selects the type of encoding scheme (depending of the received query command parameters), FM0 logic and Miller logic blocks ensure the data coding. Preamble fields are stored in four registers and will be selected using a multiplexer. The logic control block organizes the transmission of the appropriate preamble, data and end of the frame.

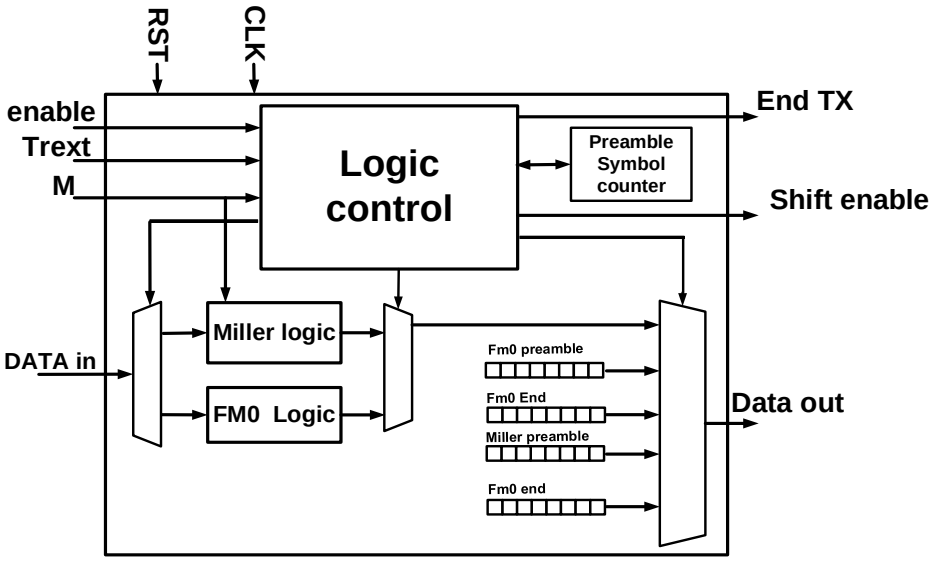


Figure 4.9: RTL schematic of FM0 and Miller encoder

4.2.1.2.5 PRNG (Pseudo random number generator):

A Pseudo random generator (PRNG) produces a series of numbers which are statically independent. In EPC C1 Gen 2, the PRNG has the following properties:

- The probability that any pseudo random value shows up as the next output is between 0.8 ×2-16 and 1.25×2-16.
- The probability that any two or more tags generate simultaneously the same 16-bit numbers sequence is less than 0.1%.
- The probability to predict the next Pseudo-random number from previous output is less than 0.025%

The architecture of the PRNG is very important, and especially for tag communication security. There are a few proposals for PRNG in the literature attempting to fulfill the EPC C1 Gen2 requirements. Table 4.2 shows some of these PRNGs.

Table 4.2: Pseudo random number generator

Algorithm	**Output size**
AKARI1A[Kalikinkar2011]	16
Mandal et al[Honorio2011]	16
LAMED [Lopez2008]	16

The authors of LAMED [Lopez2008] showed that the implementation of their PRNG is compliant to the EPC C1 Gen2 standard requirements [EPC2008], and requires a low area (1.5k gates). This low area is acceptable for a tag implementation. In this work we have chosen to implement this design [Lopez2008].

4.2.1.2.6 Slot counter

To avoid response collisions, EPC C1 Gen2 has introduced the slot counter and the Q algorithm [EPC2008]. This module is responsible for calculating the slot number for which the tag is going to reply. Slot number calculation is based on the Q-field transmitted by the Reader in the Query command frame. Every time the tag receives a query command it picks a new 16-bit random value from the PRNG output.

```
Begin
Get Q value from query command.
Generate a pseudo random slot
If command = Query rep
Slot counter = slot counter -1
Else
Slot counter = slot counter.
End if.
End
```

Figure 4.10 : Pseudo code for slot counter implementation

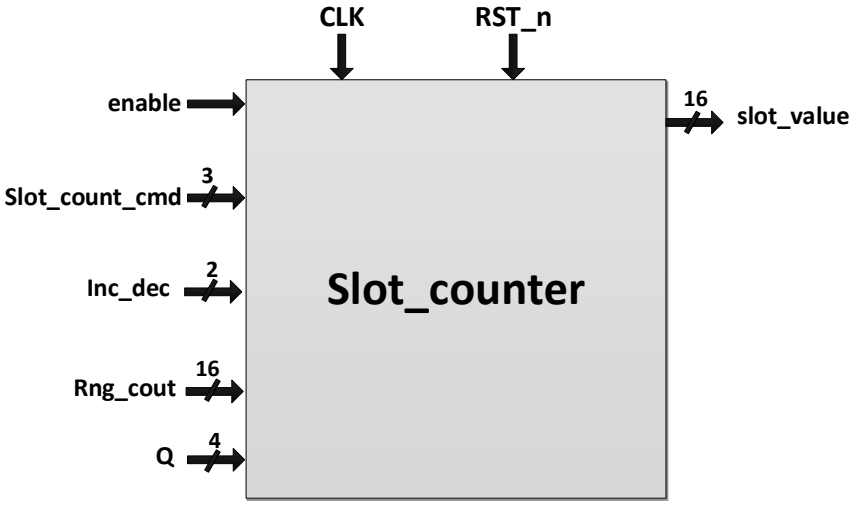


Figure 4.11: Slot counter

Figure 4.11 shows the inputs and the outputs of the slot counter block. Slot counter inputs permit to modify the slot value. For example, every time the tag receives Query Rep the inc_dec input is used to decrement the slot counter value. The RNG_cout input permits to pick up a random value from the PRNG module.

4.2.1.2.7 Session and flag controllers

As explained in section 2.4.1.4 when more than one reader are presents in the range of a tag this may cause problems because these readers communicate with the tag at the same time. For this reason, EPC C1 Gen2 standard introduces flag and session principles. To implement this feature, we propose the session and flag controller shown in Figure 4.12.

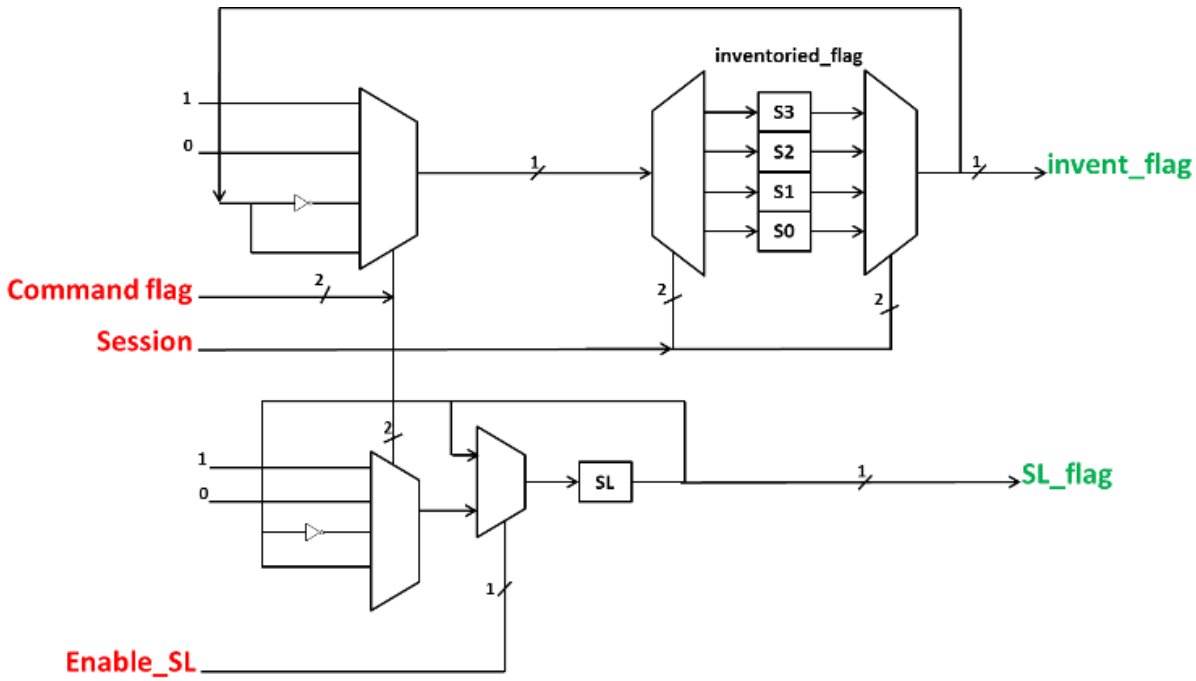


Figure 4.12: Session flag controller

The session input permits to choose among the sessions the tag participates during an inventory round. The enable_SL input permits to enable the Select flag modification [EPC2008], and the two outputs invent_flag and SL flag hold the current used flag during inventory round.

4.2.1.2.8 Tag memory

As described in figure 2.4.1.5 EPC C1 Gen2 specifies four memory banks, reserved memory banks, EPC memory bank, TID memory banks and user memory banks as shown in Figure 4.13

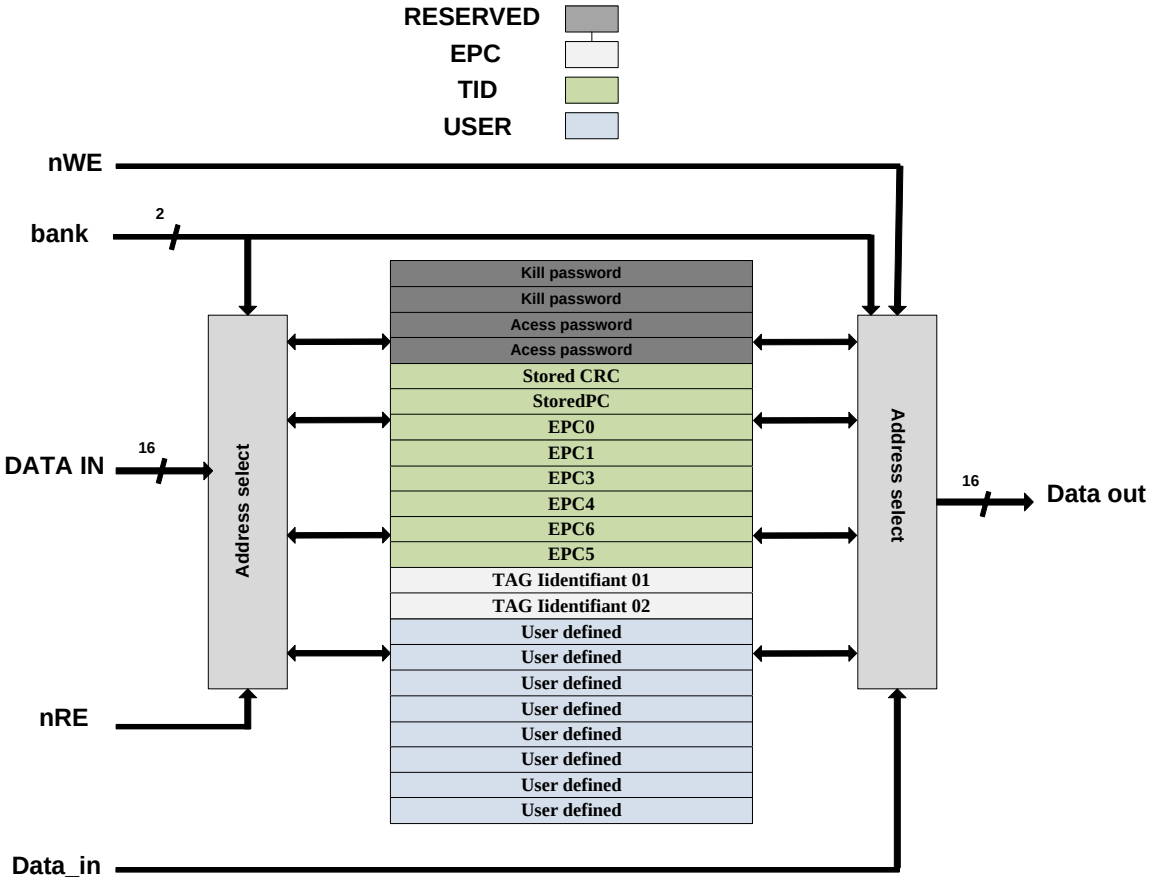


Figure 4.13: Schematic of DATA memory module

In EPC C1 Gen2 data is accessed in 16 bit words. As shown in the Figure 4.13. The two pins nRE and nWE are used to distinguish between memories reads and writes. For simplicity we use the FPGA SRAM memory to emulate the tag memory. This memory addressed by 5 bits plus 2 bits to choose the memory bank.

4.2.1.2.9 Backscatter clock generator:

The backscatter clock generator is a configurable clock divider. It feeds the transmitter module to send data with the data rate requested by the reader. This data rate depends to the predefined TRcal and DR parameters. The TRcal and DR allow choosing the clock divider values. The counter uses this value to generate the backscattering clock. Figure 4.14 shows the pseudo code of the backscatter clock generator and Figure 4.15 shows the RTL level description of clock generator design.

```
Begin
  Store all possible TRcal values and divider N value
  Get received TRcal value from PIE decoder
  Get DR value from command decoder.
  Choose the divider N
      N= f (DR,TRcal)
Loop: Increment Counter
if counter= N/2

   output =1

if counter =N

   output =0

Return to loop.

End.
```

Figure 4.14: Pseudo code for backscatter clock generator design.

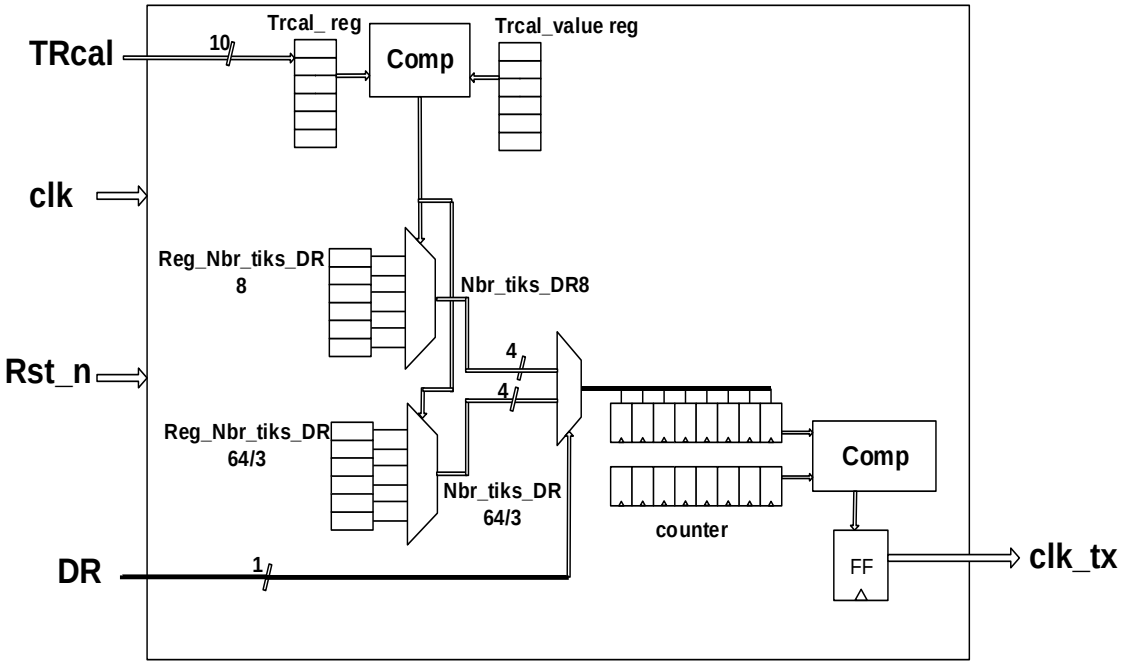


Figure 4.15: Backscattering clock generator

4.2.1.2.10 Timer T1 and Timer T2

EPC C1 Gen2 [EPC2008] specifies two response time-out T1 and T2. Timer T1 and timer T2 are two counters to measure the time T1 from reader transmission to tag response, and the time T2 from tag response to reader transmission. The computations of T1 and T2 durations are based on RTcal and TRcal parameters.

4.2.1.2.11 Backscatter generator

The backscatter generator schematic is shown in Figure4.16. This module controls the construction of a tag response packet and controls all transactions between FM0 and Miller encoder and the finite state machine module. Backscatter generator is responsible for:

1. Creating an EPC C1 Gen2 compliant packet and passing it to the FM0 and Miller encoder for conversion to FM0 or Miller encoding
2. Adding the preamble at the beginning of any packet
3. Activating the CRC16 calculation for the frame that will be sent.
4. Defining the start and the end of a data packet.

The blocks (FM0 & Miller, CRC16) provide a done indicator, which helps the generator to organize the construction of the actual packet.

The state machine module uses internal control sequence to indicate to backscatter generator which type of packet response must be sent.

4.2.1.2.12 Finite state machine

According to EPC C1 Gen 2 the tag transitions from one state to another is done when receiving commands from the reader. The tag must implement seven states as specified in [EPC2008]. The transition between these states is done during the Select, Inventory and Access operations. These state are detailed in chapter 2. We have designed a FSM module to control the tag state transitions. The module creates response packets and transmits it to the backscatter generator for encoding.

4.3 RFID tag validation

4.3.1 *RFID Emulator Functionality Validation*

Design validation is crucial. It reduces the risks and the costs by enabling early detection of errors. A RFID tag development includes software and hardware parts. The first part is the specification of the RFID tag architecture (discussed in this chapter). Then the design flow consists in several stages and involves many development tools to ensure simplicity, modularity and efficiency of accomplishment of design.

The design flow shown Figure 4.16 includes the following steps:

1. Development of RFID tag VHDL model and synthesis using FPGA development tools
2. Validation and functional verification using simulation (test bench)
3. Placement & routing phase, then a second simulation is performed.
4. Implementation of the circuit on FPGA
5. Signal extracted from the Analog RF Front-end (AFE) are fed to the FPGA allowing testing all tag functionalities. In this implementation our tag perfectly answers to all the RFID commands.

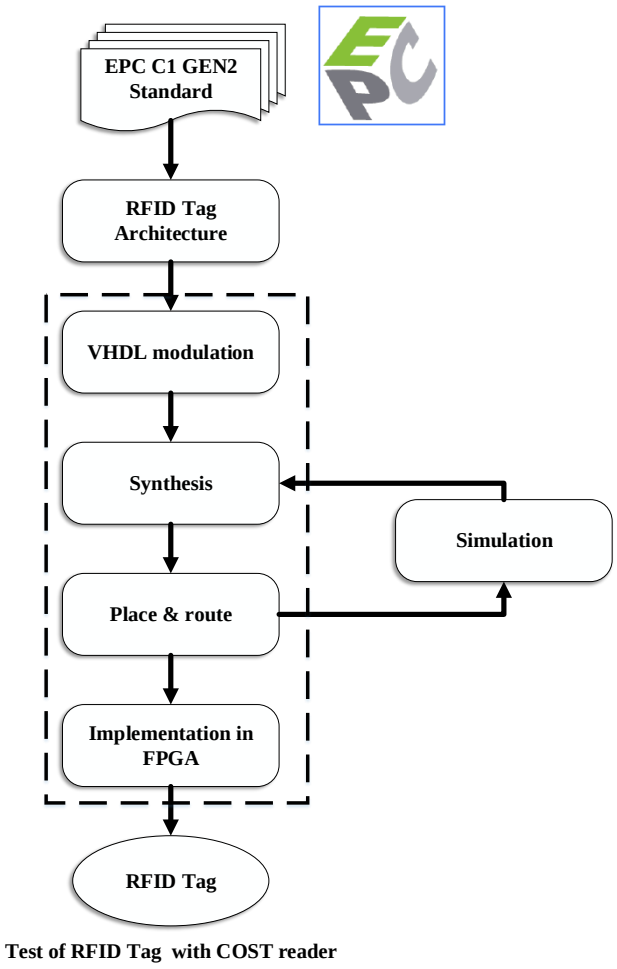


Figure 4.16: RFID tag design and verification flow

The modularity of the architecture has simplified the verification operations. The modules have been tested during the development. The test-bench of the different modules have been developed in two complementary ways:

1. Unit test-bench:

For each module a unit test-bench has been used. For example, Figure 4.17 shows the results of the state machine module functional simulation. The FSM module switches between all the possible states as described in the last section.

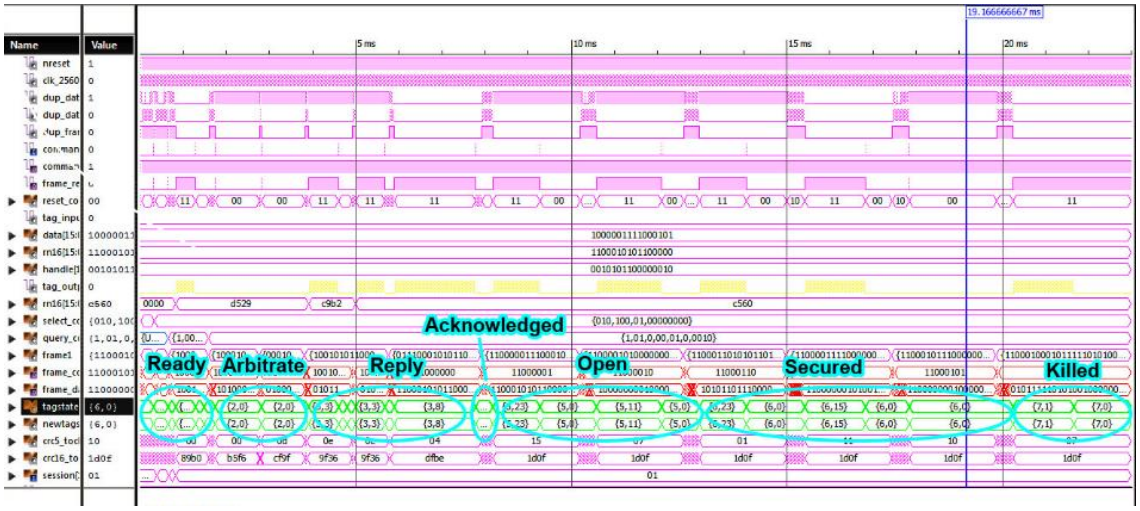


Figure 4.17: Finite state machine functional simulation with a unit testbench

2. Global test bench:

As shown in Figure 4.18, we have also developed a fully compliant EPC C1 Gen2 RFID reader model. This reader model is totally written in VHDL Using this model we are able to send all EPC commands to do an inventory round as a real RFID reader. This allows us to validate by simulation all the tag capabilities. We have also added to this RFID reader model a functionality to display error signal messages which report tag operation errors and helps us to find their location.

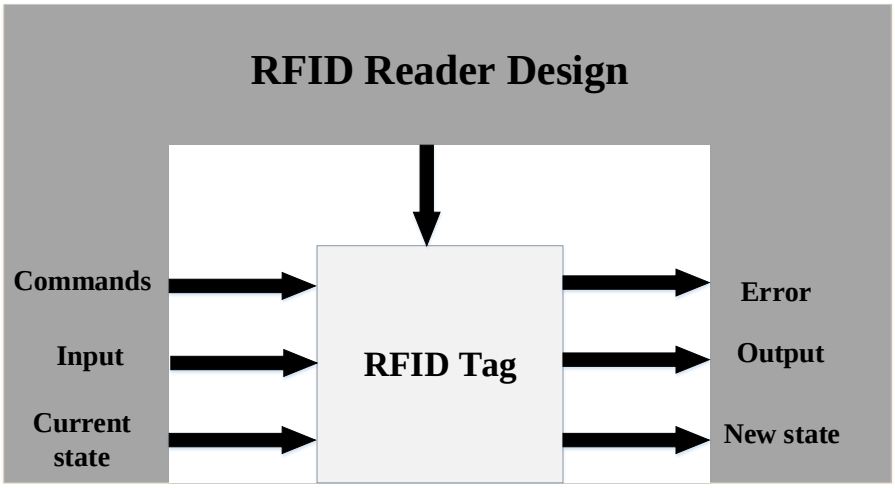


Figure 4.18: RFID tag Global test bench

4.3.2 *RFID Emulator Validation in UHF RFID Test Platform*

After the validation of the RFID tag digital part, the emulator has been validated within a real RFID environment. Figure 4.19 shows the experimental validation of the UHF RFID tag emulator. The validation has been carried out with an UHF RFID protocol analyzer. The used UHF protocol analyzer allows sending different EPC C1 Gen2 commands and displays the tag responses as shown in Figure 4.20. The response results have been analyzed and verified against the EPC C1 Gen2 standard.

The UHF protocol analyzer allows tuning all the communication parameters used to communicate with the emulator. Thus, the emulator has been validated with different communication parameters within the range of the standard. This important validation step allows us to check the emulator fulfills the communication timing specifications of the EPC C1 Gen2 standard. The tests and measurement setup are shown in Figure 4.19. The RFID tag is placed in an anechoic chamber at a fixed distance of 2 meters from the reader antenna. The RFID protocol analyzer in Figure 4.19 contains three parts.

- **The radio frequency part:** this part contains a Quadrature ASK modem modulations
- **The digital baseband part:** this digital part is designed on a FPGA and provides the interface between the software part and the radio frequency part. It allows the execution of all the commands sent by the software part to build the frames and to send it via the RF part and also to decode and analyze all the received frames sent by the tag under test (i.e. here the emulator).
- **The software part:** this part is designed using LabVIEW. It allows configuration of all parameters and frames of an UHF RFID reader such as timing of the Tari, PW or RTcal parameters. It defines the encoding schemes (FM0 or Miller) that the tag should use for the response. This LABVIEW interface is thus essential to define the test to be carried on the tag. It also provides the visualization of the tag response signal as shown in Figure 4.20.

Figure 4.20 shows a validation sequence which validates the functional behavior of the tag. The tag answers to all inventory round commands. During this experimentation, we have also verified that the tag fulfills the timing specifications of EPC C1 Gen2 standard [EPC2008].

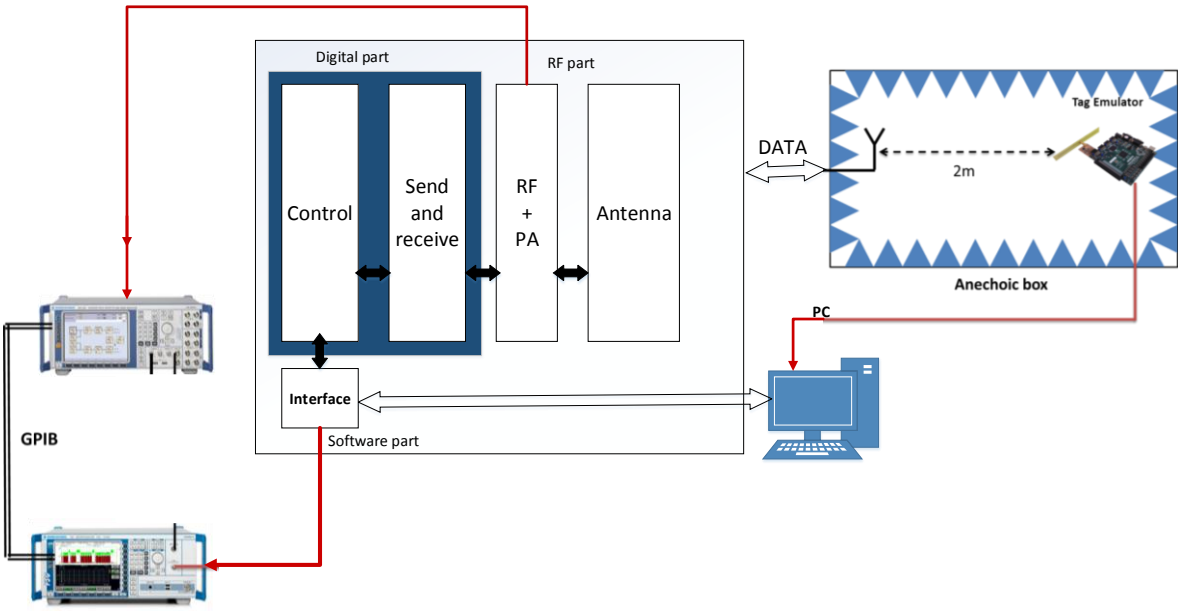


Figure 4.19: Tag validation using the protocol analyzer

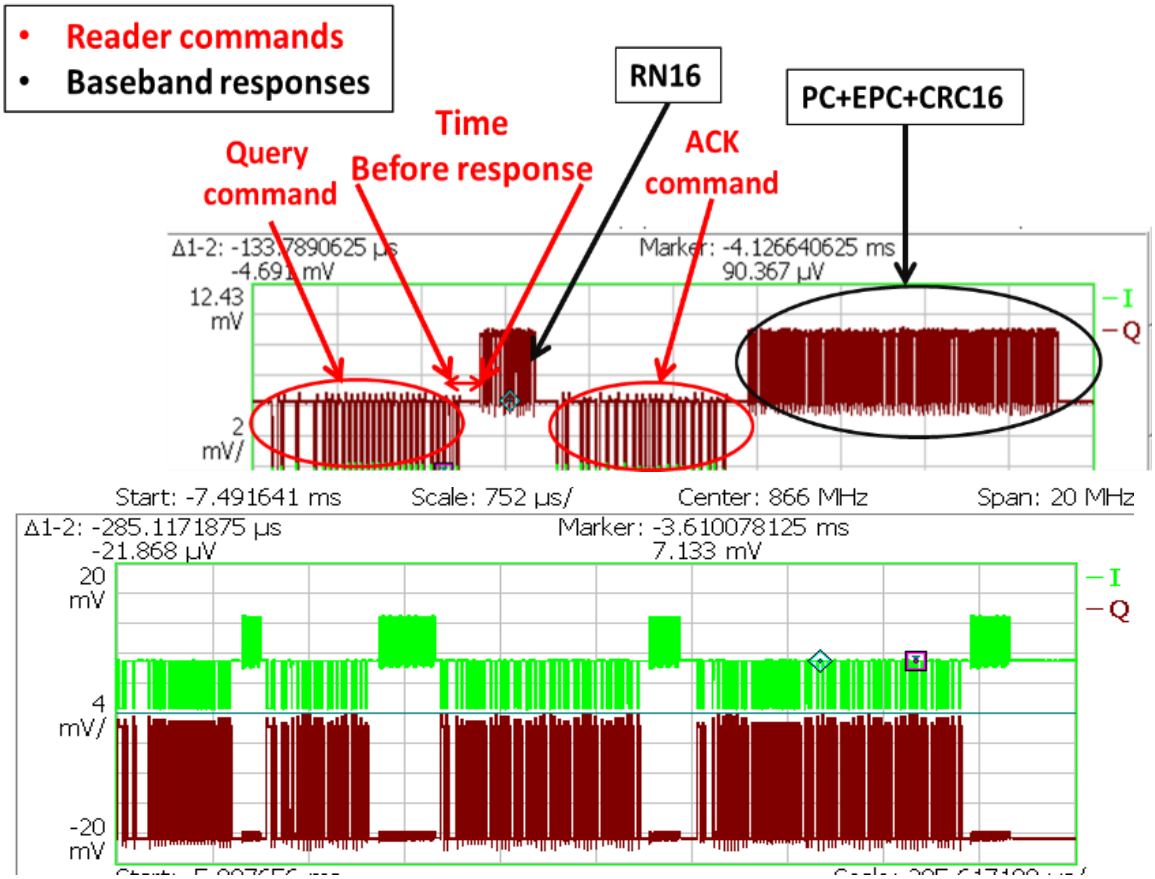


Figure 4.20: Experimental results of UHF RFID baseband response

4.4 Architecture and system descriptions of the fault emulator for RFID System (RFIM)

Analyzing aberrant functioning which may result from a faulty tag is a complex task, and a time consuming process. It is important to perform this analysis at the beginning of the design process. Fault injection is necessary to analyze the RFID system robustness and its countermeasure efficiency. To perform this fault injection it is necessary to have specific mechanisms which control fault injection and provide information on error propagations.

Such analysis is necessary to decide what is the best fault tolerance strategy (the countermeasures to implement and the parts of the circuit to be protected). In this context, two complementary aspects are considered as part of this work: the fault injection and the analysis of results obtained after the injection campaign. The principle of fault injection is to compare the golden circuit behavior (without fault injection) with its behavior in the presence of injected faults during a communication with the reader.

The main objective of this thesis is the development of a methodology and of a fault injection platform that will be used to improve the robustness of a UHF RFID tag. The fault injection platform is based on the hardware emulator described in the last section of this chapter. This emulator will be instrumented for fault injection and error analysis.

We focus on the analysis of RFID digital IC taking into account the most important aspects of the fault injection platforms which are:

- The methodology followed to ensure controllability, observability, repeatability, reproducibility of experiment
- The way to collect and measured data during experiments
- Fault injection management (time space, type).

This environment is based on the FARM [Arlat1992] model attributes previously described in the state of the art. We present the assumptions (in terms of FARM model described in chapter 3) and choices that underlie the organization of the RFIM platform.

4.4.1 *Set F*

It is all the faults to be injected into a fault injection campaign, the default model to be selected is transient single bit flip, since it is very similar to the faults occurring in real systems. Every fault will be characterized by:

- Fault injection time
- Fault injection location

Therefore, each fault corresponds to flipping a single or several bits in a UHF RFID baseband registers.

4.4.2 *Set A*

In this work, we determined in advance where the fault should be injected, based on the analysis of the system and the communication protocol, we locate all RFID tag sensitive registers which can generate error and disturb the behavior of the RFID tag.

4.4.3 *Set R*

This set defines all information that will be obtained by observing the behavior of the tag for each fault injection experiment. This allows us to identify differences with respect to the fault-free system behavior. In RFIM environment all operations of the tag are monitored by a processor that compares the state of the faulty tag with the state of golden tag, and the output result of this comparison.

4.4.4 *Set M*

At the end of the fault injection campaign, a report will be generated, this report concerning the dependability measures, and fault coverage computed on the whole Fault List. Faults are classified into one of the following categories:

- No output errors: without produced error, in this case the tag responds to commands of reader and there no error effect.
- Fail-Silent Violation behavior: in this case the fault is not in the most sensitive register, but there is downgrade in RFID tag performance, and the tag responds to reader with incorrect results.

- Fault with output error: in this case the tag does not respond to any Reader command because the faults are in a sensitive register and produce a serious failure.

4.4.5 *FARM set of RFIM platform*

As presented in Figure 4.21 the RFIM platform is divided into eight modules: interface monitoring, fault injector, activation of injection, event detector, golden and faulty tags, register comparator and embedded microprocessor. The host computer permits to print RFIM results.

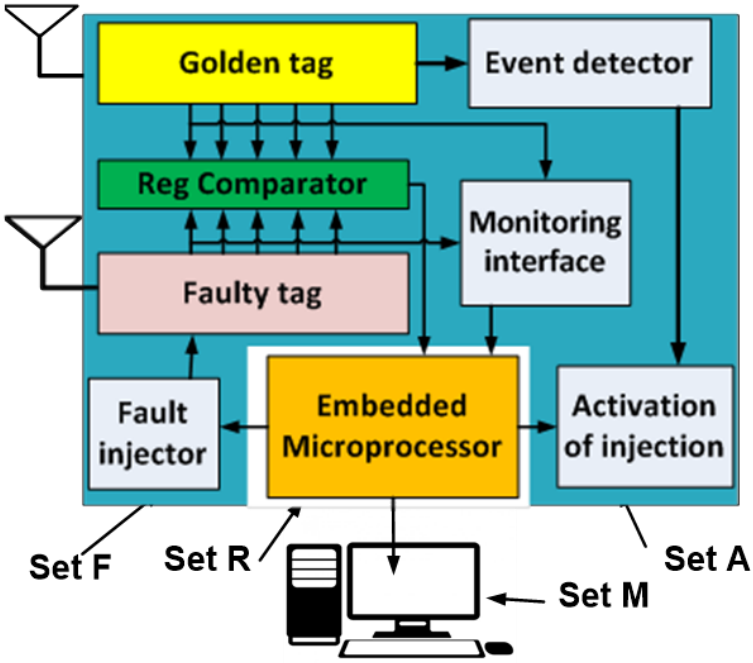


Figure 4.21: RFIM platform for faults injection and monitoring

The embedded microprocessor controls all the platform modules and then permits to perform on-line tag monitoring. The processor also allows the on-line capture of data in the two tags basebands, this allow analyzing the RFID communication. The interface monitoring is a mechanism that transports the internal register values from the tag basebands to the microprocessor. This interface monitoring block uses a First-In-First-Out (FIFO) memory in order to compensate the latency of the microprocessor to outputting register values. Faults are only injected in the faulty tag and the golden tag which is always fault free is served as a reference. The register comparator compares all the internal registers of the golden and the faulty tags. This comparison helps the embedded processor to detect and to localize faults and errors in the faulty tags.

The activation of injection module helps the embedded processor to choose the fault injection times. The event detector permits to trigger fault injection time with specific tag FSM states or specific values of the tag internal registers. This block is designed by the user according to the target of the fault injection time.

The on-line monitoring of the tag helps to understand the successive operations of the RFID system. According to the EPC C1 Gen2 standard, the successive tag states depend on the commands sent by the reader. The monitoring of the state and register values helps designers to understand the behavior of the tag in a faultless or faulty environment. The processor allows designers to read, to store and to edit the content of all the internal registers of tag. These registers are accessible through the interface monitoring block that is connected to the microprocessor.

Figure 4.22 shows an example of log file generated by the RFIM emulation platform during communication between RFID Reader and tag. This file is printed on the host computer monitor. This log file helps to analyze the behavior of a tag in the presence of fault. In fact, it helps to analyze the execution of EPC C1 G2 protocol and to monitor the fault effects on each part of the faulty tag.

The log file in Figure 4.22 shows that the emulator can monitor all the communication parameters exchanged between tag and reader. These parameters concern both the tag functional behavior (names of the received commands, for example QueryRep. or tag state, for example Ready) and the communication characteristics (for example, TRcal ,RTcal parameters [EPC2008] The log file also indicates the presence of fault (fault injected: no).

```
1E240 : Tag state is    : Ready
1E740 : TRCAL value is : 123
1EC40 : RTCAL value is : 166
3D880 : the command is : Query_rep
7B100 : the slot     is : 7a0d
7B600 : Tag PRNG     is  : fa0d
F7C00 : Fault injected : no
F8C00 : Tag state is    : reply
```

Figure 4.22: Log file for communication analysis

4.4.6 *Fault injector*

To perform the fault injection, we reuse and adapt a fault injector proposed in [Mezzah2005] [Lala2012]. The proposed fault injector described in Figure 4.23 able to inject permanent or transient single or multiple bit faults on any signal (vectors or single bits) in combinational and sequential parts of the HDL descriptions. The fault injection operation on a specific internal signal is made by inserting a Fault Injection Block between the signal source and its destination in the original description.

The fault injection system consists in two main parts:

1. The fault injection controller
2. The fault insertion logic as described in Figure 4.23.

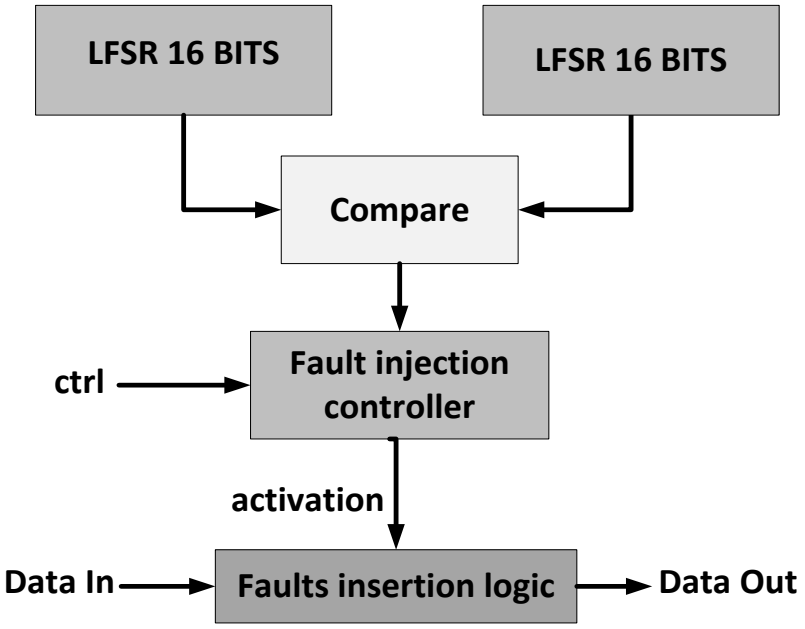


Figure 4.23: Bloc diagram of the fault injector

The injection controller controls the insertion of faults using ctrl configuration vector (provided by the fault injector user):

- Ctrl = 0000: injection of permanent faults.
- Ctrl = 0001-1111: injection of transient faults.
- When ctrl=0000, as shown in Figure 4.24, the fault injection logic injects Stuck-at '1' or Stuck-at '0' depending on StuckAtValue vector. These faults are only injected in the bits at '1' in the FaultAtBits vector.

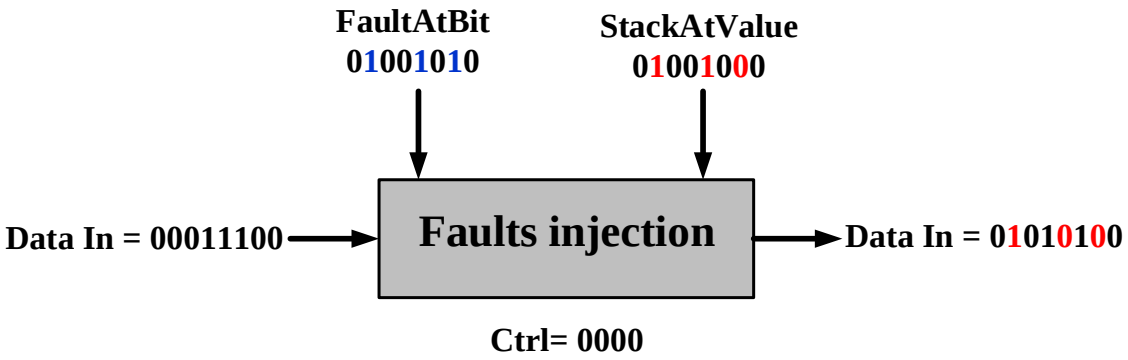


Figure 4.24: Permanent fault injection

As previously explained, in the case of permanent fault, bit locations where the fault will be injected are configured by the user, and in the case of transient faults injection the controller rotates (for each clock cycle) the FaultAtBits and StuckAtValue vectors in order to inject faults in random location. In the case of permanent faults injection, the faults are directly injected as soon as the value of ctrl equals 0000. But, in the case of transient faults injection, the time for injection is random, in addition to random location fault injection, it is also necessary to provide a mechanism for a random timing fault injection.

The controller determines these timings thanks to two pseudo-random generators as shown in Figure 4.25. We use two Linear Feedback Shift Registers (LFSR) with 16 Flip-Flops to implement two pseudo-random generators. The 16 bits LFSR allows the injection of transient faults in intervals compliant with RFID system operations.

The two LFSR concurrently generate different sequences and their results are compared to determine if there is faults injection. The ctrl value, ranging from 1 to 15, corresponds to the number of bits of the two LFSR that are compared. If all the bits compared are equal, then the fault injection is activated. Thanks to this mechanism the ctrl value controls the rate of the fault injection.

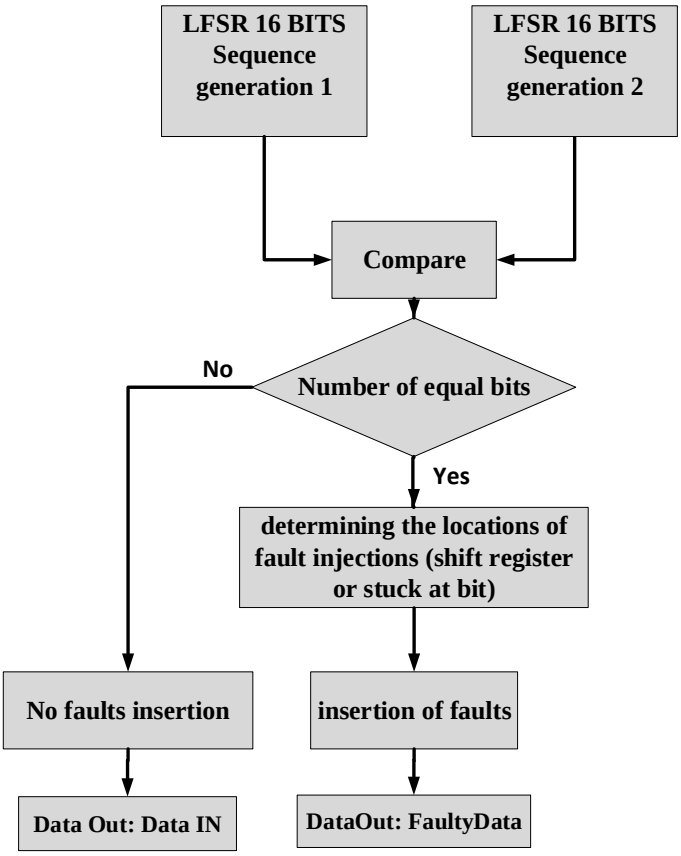


Figure 4.25: Flowchart of the transient faults injection

The theoretical rate of the fault injection with ideal random generators is given in Table 4.3.

Table 4.3: Fault injection rates given by ideal random generators

Ctrl	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
%	50	25	12,5	6,75	3,37	1,68	0,84	0,42	0,21	0,10	0,05	0,25	0,0125	0,006	0,003

For example, if ctrl= 0001, the fault will be injected if the first bit of the two LFSR are equals. Among the four possibilities (00,01,10,11) there are two cases (00 and 11) that activate the fault injection. This corresponds to the probability of 2/4 = 0.5 (50%) given in the first column in Table 1. In our pseudo random faults injection mechanism rates are different due to the non-ideality of the 2 pseudo-random generators. The experimental results obtained by simulation with the pseudo-random generators are given in Table 4.4. In Figure 4.26 the fault injection rates with ideal and non-ideal random generators are compared. This shows the low impact of using pseudo-random generators for performing fault injection.

Table 4.4: Pseudo-random faults injections rates obtained by simulation with 10000 cycles

Ctrl	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
%	62,65	34 ,6	17,89	9,16	4,69	2,37	1,13	0,57	0,28	0,15	0,064	0,38	0,015	0,009	0,006

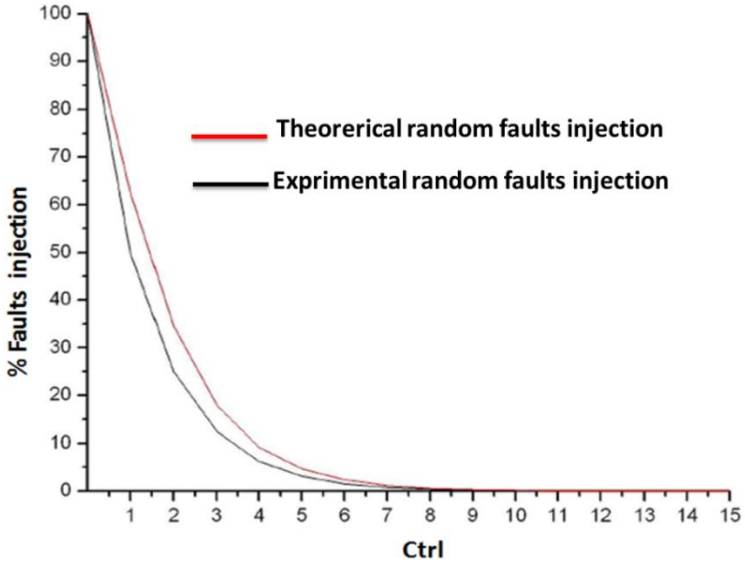


Figure 4.26: Fault injection rate depending Ctrl

4.4.7 *Fault injection mechanism:*

To illustrate the proposed fault injection mechanism within the digital baseband we detail a fault injection operation on a given example.

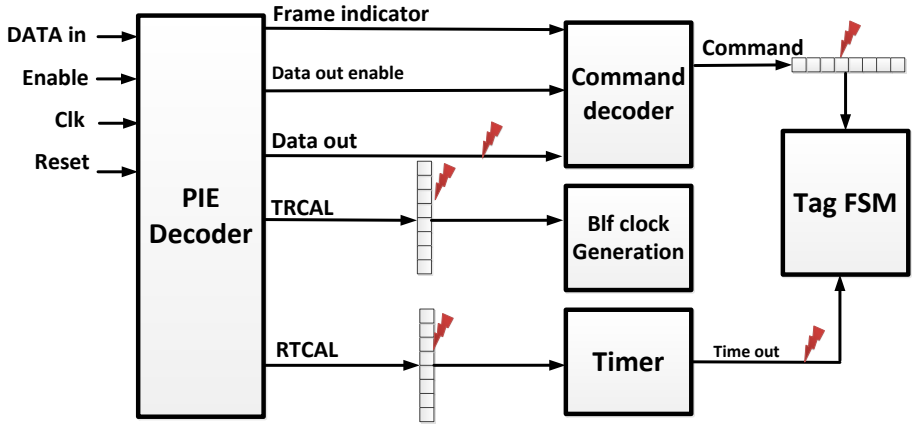


Figure 4.27: Fault injection in tag registers and propagation

Figure 4.27 shows the interconnections between different blocks in the tag architecture, the PIE decoder decodes the symbol sent by the reader and computes some communication parameters like TRCAL, RTCAL [EPC2008], Data_out which are then processed or stored in

the digital baseband. For instance, the TRCAL data is stored in register within the Timer block (this data is kept in a register since it is used during all the communication), while the data_out data is passed to another block which computes it immediately. The command decoder converts the Data_out to a specific command stored in a register which will be used by the tag finite state machine (tag FSM). Thus, in some parts of the design, the faults can propagate from combinational blocks to others or from registers to others and then disrupt the functionality of the tag, as shown in Figure 4.27To illustrate the propagation and the effect of a fault, we show in Figure 4.28 the effect on the tag FSM of a fault impacting the command register. In this case the state machine may be disrupted, and may perform another unwanted operation. This is one of many possible scenarios for attacks by fault injection. In Ready state, the tag should receive commands to authenticate the reader and to calibrate several parameters used during response. Due to the fault injection, FSM directly goes to a secure state that permits to read the memory without authenticate the reader. In other cases the FSM may be blocked in one state, and the tag will never respond until it has been powered off.

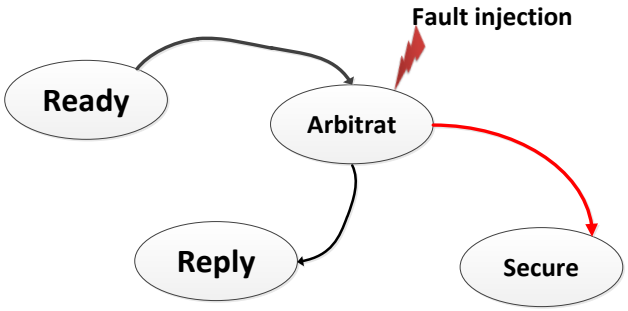


Figure 4.28: Faulty transition in the FSM due to fault injection

The developed fault injector allows configuring many faults types and controlling the fault injection rate. The Figure 4.29 illustrates the fault injector interconnection with the targeted component in digital baseband circuit.

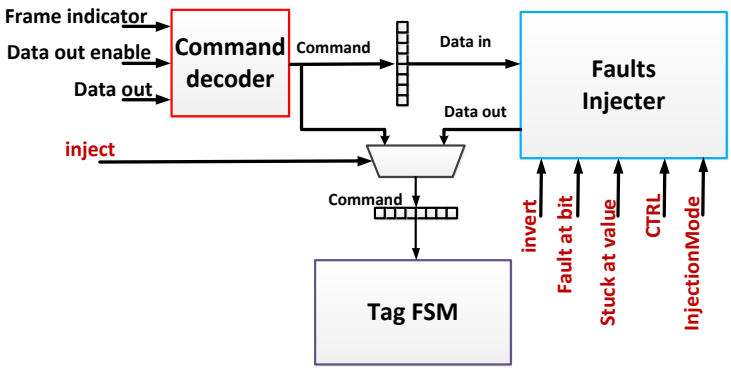


Figure 4.29: Register Fault injection

In Figure 4.29 the command register is impacted by the fault injection. The inject input is used to enable or disable the faults injection. A multiplexer controlled by this signal selects the faulty data or the fault free data of the command register. The invert pin is used to inject stuck-at or bit flip. The inject_mode pin is used to configure the bit location of the fault injection. This location can be randomly chosen or can be a fixed location. For fixed location we use the FaultAtBits input to control the bit position and Stuck-at-value input to configure the type of stuck-at injected in the selected position.

The timing diagram in Figure 4.30 shows a simulation of the faults injector for an 8 bits register. The CTRL input to select the fault injection rate is set the value 0010 to obtain a 35% fault injection rate as shown in Table 4.3. All bits of data_in are equal to 1 during the fault injection. From cycle 1 to 6, we set the invert signal to make bit flip fault injection. From first to 4th cycle, we select a random position injection mode by setting the injection mode input to 1, and after the 5th cycle the random position mode is disabled and the injection become fixed. Thanks to the Fault_at_bit vector, as shown in Figure 4.30, the faults were injected into the bit 0 and bit 7 which are defined by the Fault_at_bit vector value (10000001). By disabling invert input we change the fault injection from bit flipping to stuck-at-0 or stuck-at-1.

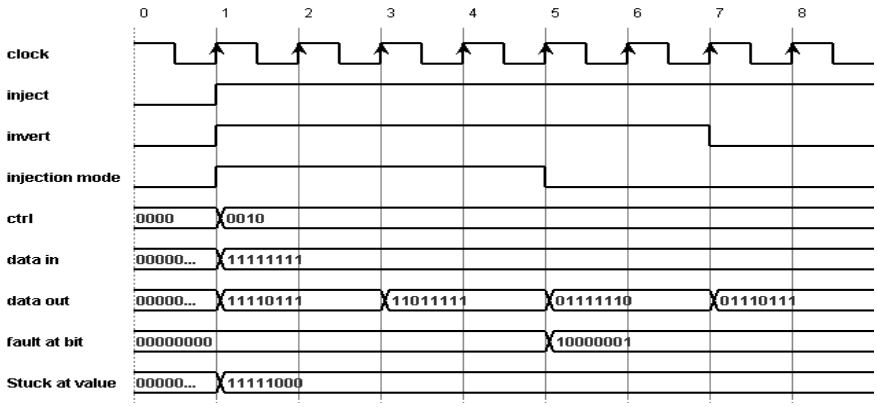


Figure 4.30: Transient Fault Injection Simulation

4.5 Experiment using RFIM platform

We are using a third party reader [AMS2014] 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. In this experiment 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)

The main EPC C1 Gen2 parameters have been detailed in the previous section, we inject SEU only in the registers storing these parameters in the digital baseband, for the purpose to know the effect of each parameter on the performance of the tag. 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 above. 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.

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.

4.5.1 *Fault Effect Evaluation on read rate*

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.

4.5.2 *Standalone tag Evaluation*

A first fault injection campaign has been carried out on a single tag (the emulator) in front of the reader. Figure 4.31 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.

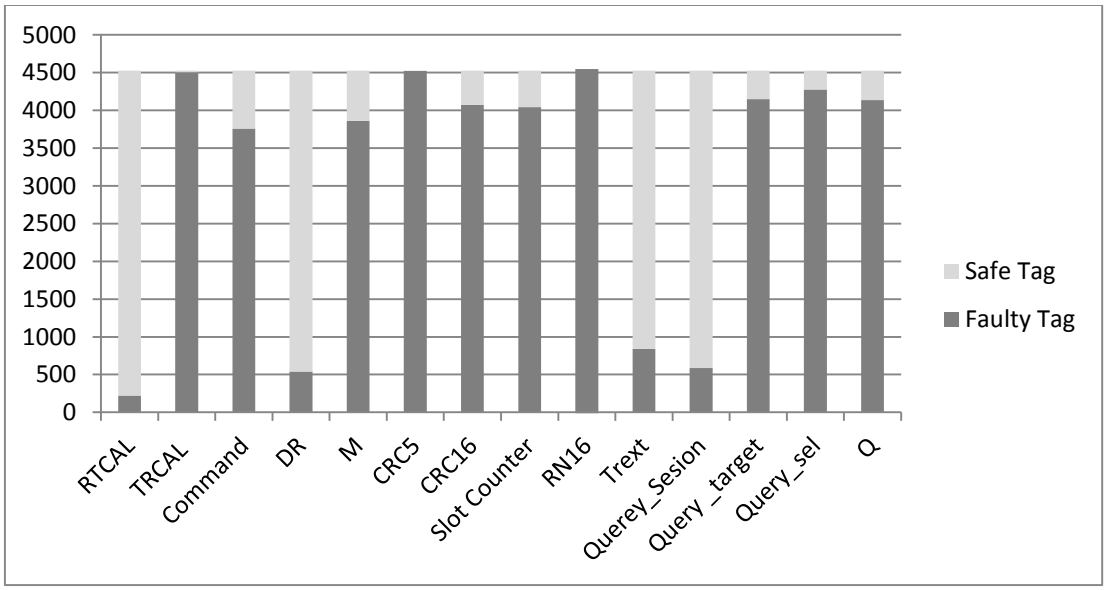


Figure 4.31: Successful tag Identifications

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 does 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 TRext 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, TRext 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.

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. Figure 4.32 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 tag1 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.

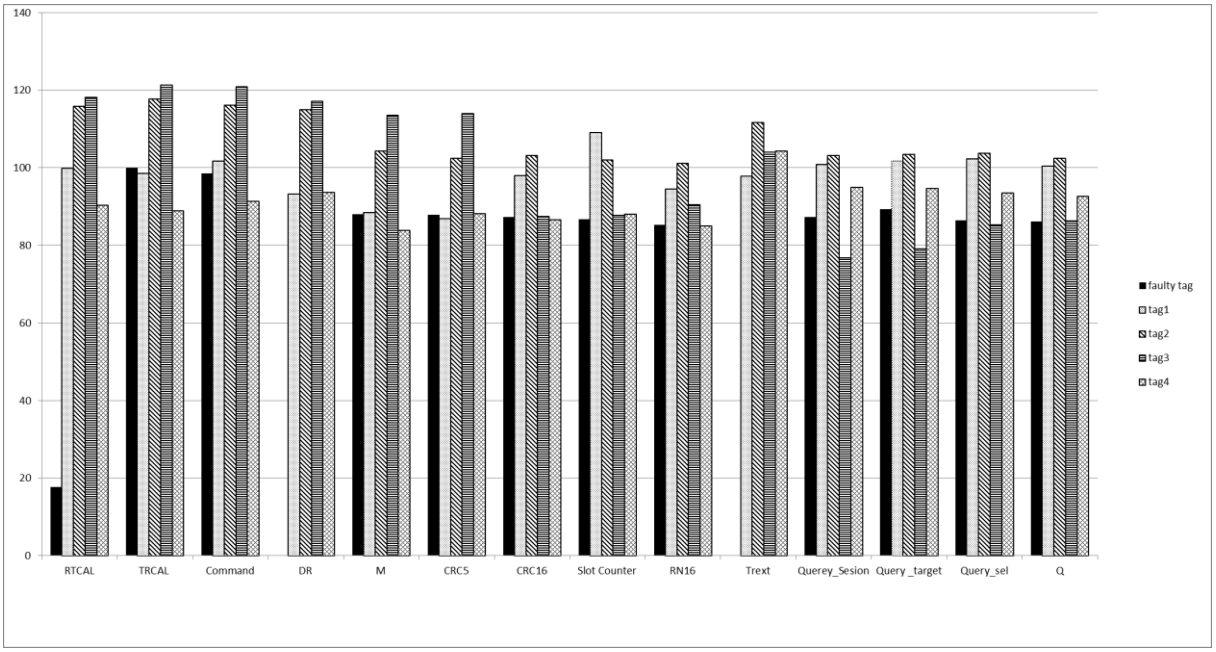


Figure 4.32: Fault effects on each inventoried tags

To know if the results depend on the software embedded in the reader we have performed the same fault injection experiment using another commercial reader.

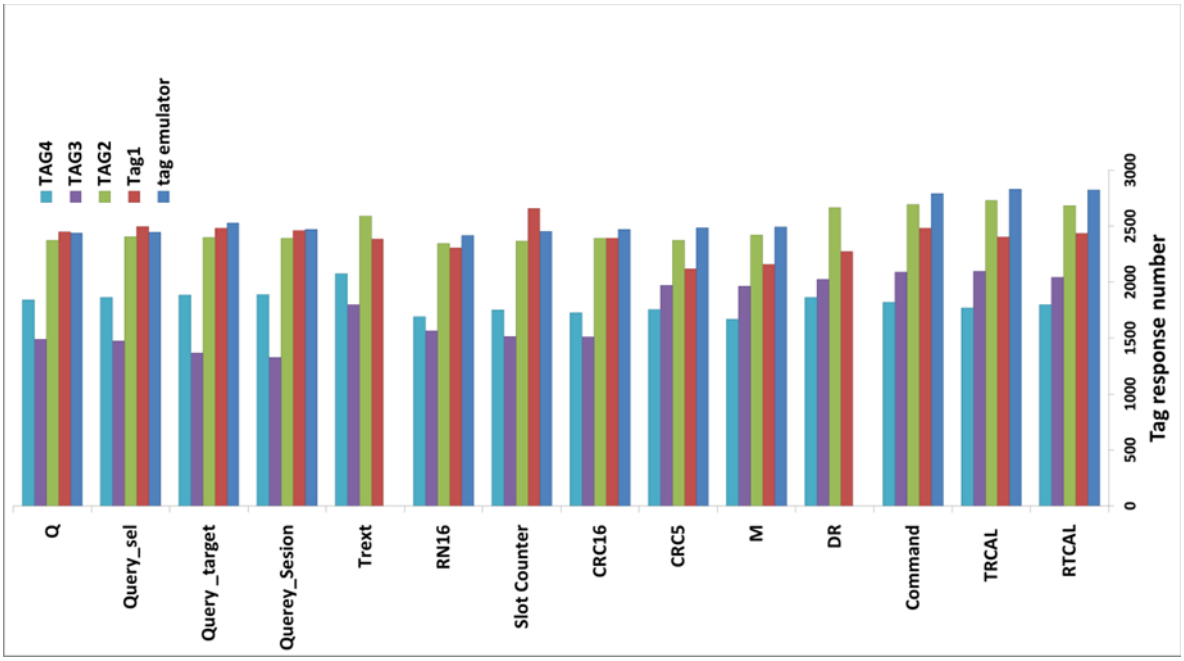


Figure 4.33: Result for second third party reader

Figure 4.33: experiment results is done with the second third party reader, we see that is the same as the results with the experiments done with the first reader, so we conclude that the vulnerability of register that stores the EPC parameter in the tag digital baseband does not depend the reader that use a different anti-collision algorithm.

4.5.3 *Fault injection effects on tag behavior*

In the first experiment we have studied the global effects of SEU on the read rate. This section studies in more details the effect of SEU on the behavior of the tag. For every injected fault, the behavior of the tag digital baseband is analyzed in order to understand fault propagation.

To facilitate the fault injection automation, we use the same microprocessor to automated fault injection and control all the experiments. The Workload generator prepares the input based on the workload library that stores all possible scenarios for fault injection. In each scenario we can change fault injection parameters. These parameters can be: fault model, fault injection time, fault duration, etc. The fault injector is responsible to inject the faults in the registers and takes the configuration from worklib. The data collector collects all the results as the number of experimentations, the injected faults, the detected and masked faults, etc. All these parameters will be shown using the monitor.

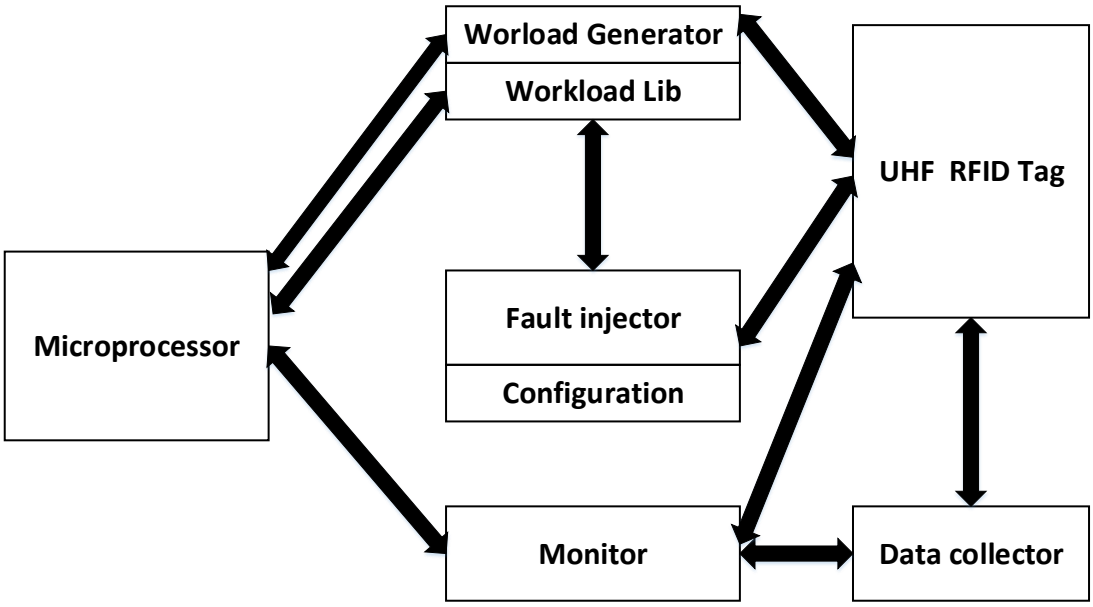


Figure 4.34: Main components of the fault injection environment 2

Diagram of Figure 4.34 shows and explains the working principle and the transaction between all the components constituting this environment for fault injection.

4.5.4 *Experiments and analysis of the fault injection data*

The experimental setup was designed and tested on the RFID tag Baseband. This section describes the fault injection campaigns: fault locations and types, fault values, timing, and the results of the data were analyzed according to several needs.

4.5.4.1 *Fault locations*

The points for fault injection are the flowing registers: TRcal, DR, RTcal, Command, TRext, M, CRC5, CRC16, Slot counter, Q, Query, session, target registers. All these registers store parameters that are used by the tag for the communication with the reader, these register have a very important role in the execution of EPC C1 Gen2 standard, 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.

TRCAL and M parameters are used to choose the type of response frame coding. (FM0 or Miller), DR and RTCal parameters are used as reference to select the frequency in which the response frame coding is done. These two parameters are also used to adjust the timing between each tag response. The role of the command register is crucial. All the commands received by the tag and that will be executed by the state machine are stored in this register. CRC register is used to eliminate erroneous packets by calculating checksum of received frame.

The slot counter register, Q and session flag are used during the inventory round. The reader uses these parameters to tune the tags and avoid the collision of responses of multiple tags.

The microprocessor chooses the faults location. For each register there is an address. Using this address the microprocessor determines which register will pass through the fault injector.

4.5.4.2 *Fault Models*

Faults injector is able to inject several faults types such as permanent and transient faults, and can determine the timing of faults injection. It can also determine the randomness of transient fault. The fault injector is controlled by the microprocessor. In this work we inject only single transient faults.

4.5.4.3 *Fault Timing*

Fault injection time is random, based on the mechanisms explained in previous section.

4.5.4.4 *Fault value*

There are a large number of choices for the corrupted value, as shown below. The contents of the register location chosen for corruption could be forced to any of the following:

- All zero state
- All one state
- Random value: a randomly generated bit vector
- Inverted state: all bits inverted
- Single bit flips in an n-bit word
- Multiple bit flips in an n-bit word

In this work and for this analysis the bit-flip has been chosen to emulate the natural and intentional faults. Fault injection is controlled by the program executed by the microprocessor. The faults occur during every inventory round.

We have improved the architecture of RFIM platform as shown in Figure 4.35. We have added some blocks to facilitate the execution of the fault injection workload. We have added memory to store the fault lists that will be injected. A memory stores instructions to manage the

fault injection campaigns. The fault injector counter gives the total fault injected by the fault injector during a campaign. The fault counter also gives us the total number of errors that generate an error response. The fault locator enables or disables the faults.

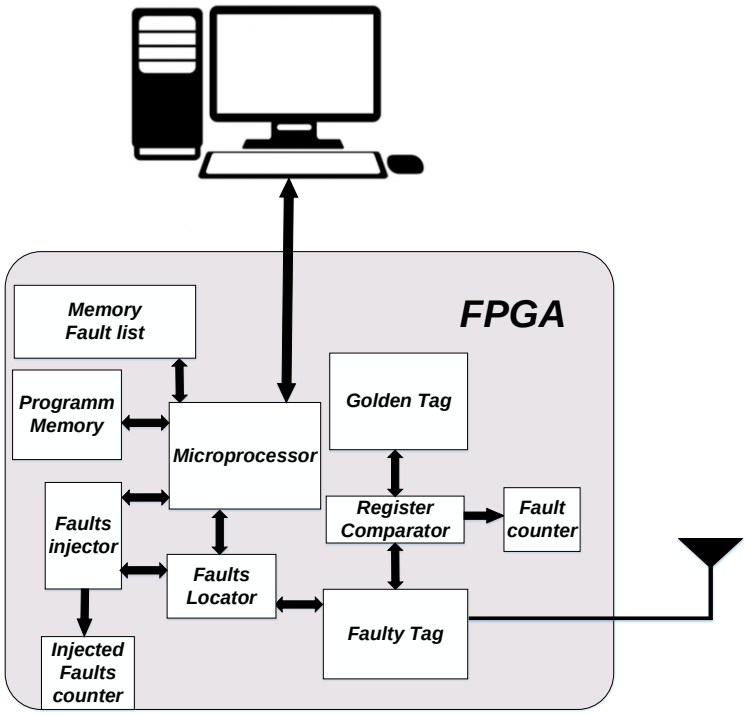


Figure 4.35: Improved RFIM Architecture

A number of response modes have been observed including:

- **Latent errors**

A latent error is an error which has not caused a detectable failure during fault injection experiments. The effect of the fault injection does not propagate because the register has been overwritten or replaced by new values before being used, or the fault was injected into a register at a time when the register is not used.

- **Detected faults**

In this case the injected fault causes an error; the error is detected by comparing between the golden run and the faulty run.

4.5.5 *Fault injection campaigns*

In this section we will describe the results achieved with 3 different workload. Then in the next section we will explain for each parameter the effect of the fault injection.

4.5.5.1 *Workload 1*

The objective of the first workload is to know the impact degree of each injection in each register of the digital baseband. We consider the error is detected if the tag does not normally continue the execution of the inventory round. Table 4.5 and Figure 4.36 show a distribution of the relative percentages of each response modes

Table 4.5: Observed response modes

	#injected faults	**Detected errors**	**Latent errors**
TRcal	11 154	9870	1248
DR	10 563	9930	633
RTcal	11 883	1461	10422
Command	10003	103	9900
TRext	10009	8300	1709
M	10894	94	10800
CRC5	11689	159	11530
CRC16	9800	50	9750
Slot	9933	633	9300
Q_query	10400	520	9880
Session	10689	10 033	656
Target	11002	1402	9600

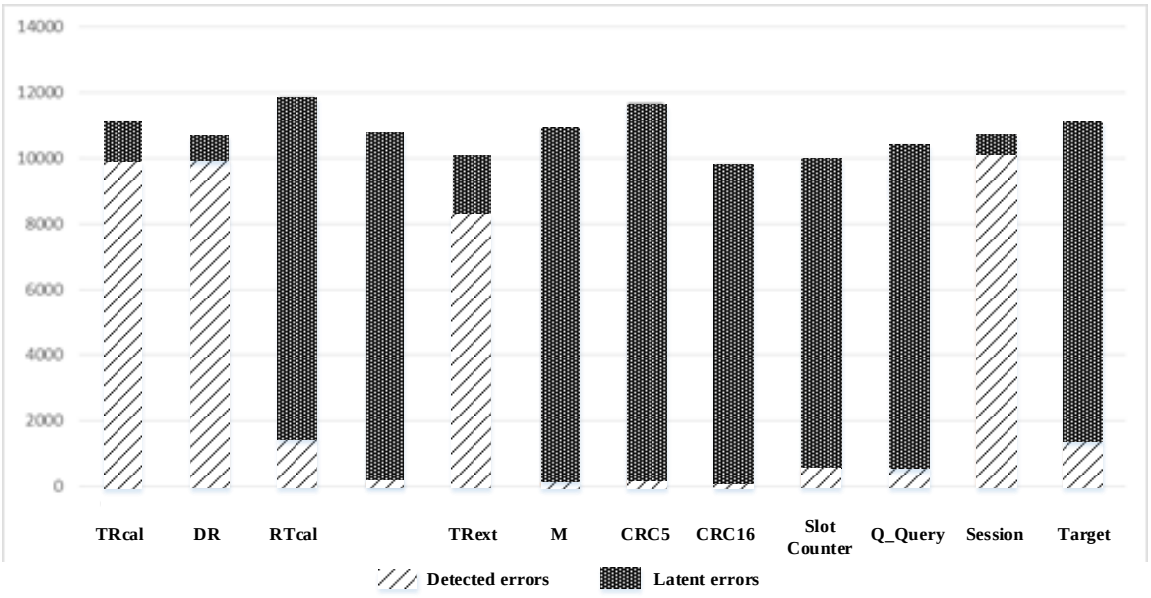


Figure 4.36: Relative percentages of each of the response

4.5.5.2 *Workload 2*

The objective of this workload is to know how the faults propagate inside the digital baseband. Based on the information path in the digital baseband, different nodes are selected and in each node a comparator is added. This comparator compares the golden and faulty tags. This allows knowing in which level one faults affects the digital baseband behavior and understanding how faults propagate and change the digital baseband behavior. Figure 4.37 shows the location of each node.

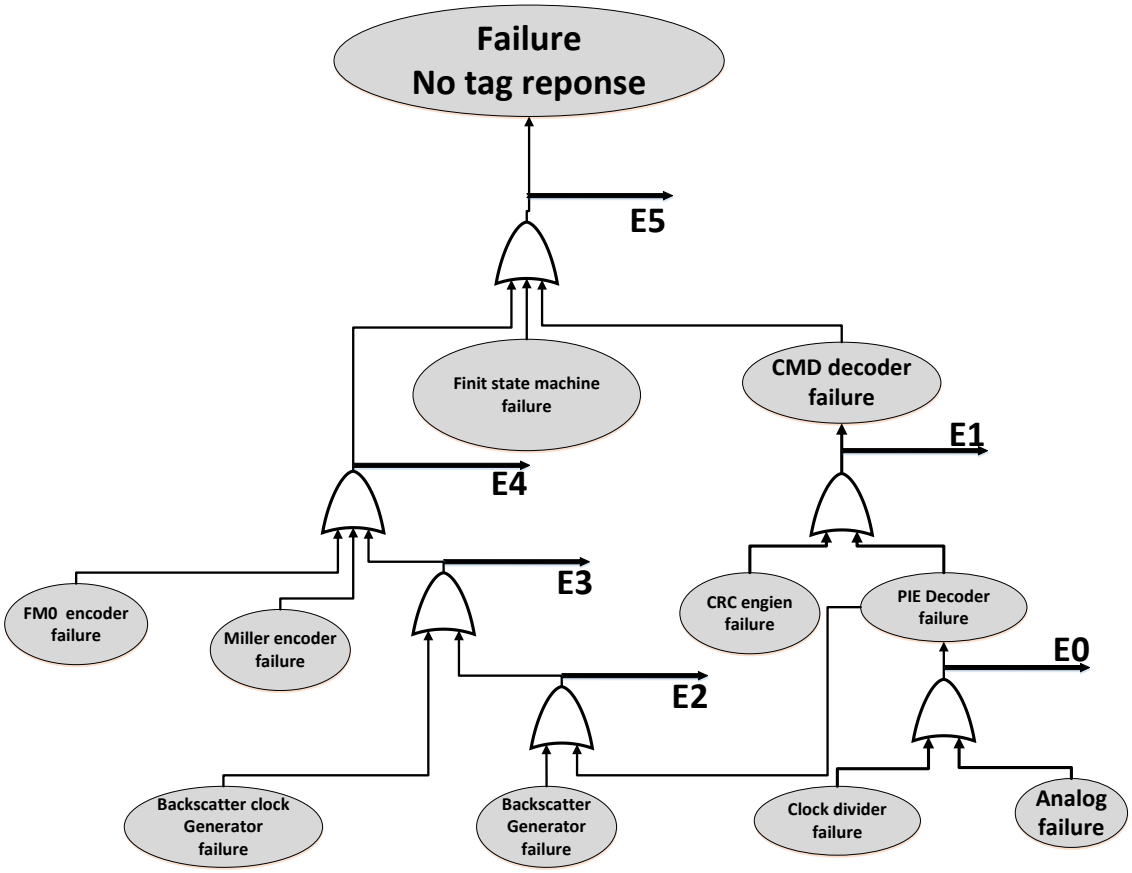


Figure 4.37: Fault tree in tag baseband

Table 4.6 shows the result of fault injection to determine the faults propagation path E_i. For each fault injection location, we look what are the effects on other blocks and the final response of the tag.

Figure 4.38 shows the path of each injected fault in the tag. The paths in red color and the grays are secure paths. These results will allow designing a robust RFID tag by protecting the weak paths to prevent the error propagation.

Table 4.6: Result of fault injection to determine the faults propagation

	E0	E1	E2	E3	E4	E5
TRcal,	0	10 000	8650	8650	8650	8650
DR	0	0	0	9670	15 000	9600
RTcal	0	13 650	0	0	0	9560
Command	0	0	0	0	0	9643
TRext	0	0	0	9400	9400	10 000
M	0	0	0	0	0	8960
CRC5	0	0	0	0	0	0
CRC16	0	0	0	0	0	0
Slot counter	0	0	0	0	0	8143
Q_query	0	0	0	0	0	9500
session	0	0	0	0	0	9650
target	0	0	0	0	9800	10 000

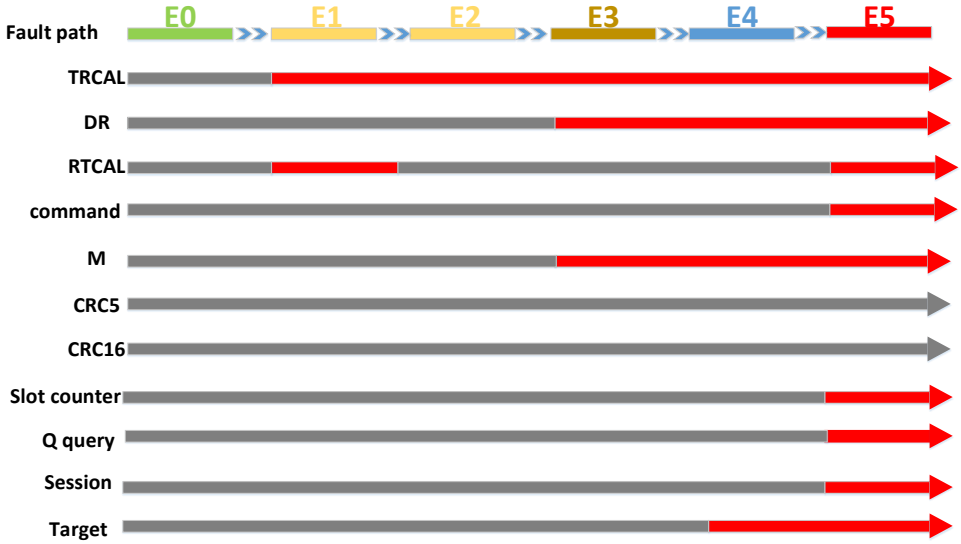


Figure 4.38: Fault path propagation

4.5.5.3 *Workload 3*

The objective of this workload is to know the effects of fault duration on tag digital baseband. In this experiment we analyze the register sensitivity to transient faults by injecting faults of different durations from 100 us to the global inventory time 544 us. The experiment is repeated for 100 inventory rounds. Table 4.7 and Figure 4.39 show the obtained results. All these results will be explained in the next section for each parameter.

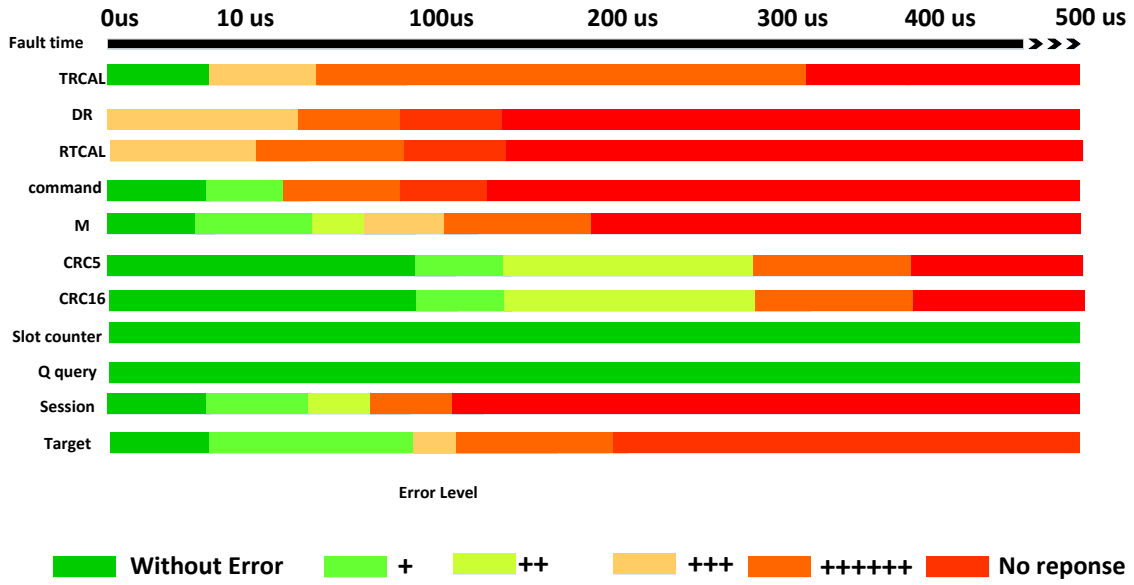


Figure 4.39: Fault duration effect on RFID tag Baseband

Table 4.7: Number of reading over the 100 inventory rounds for each parameter and depending on the transient fault duration Register fault injection effect

	5us	10 us	100 us	200 us	300 us	400 us	500 us
TRcal,	94	38	0	0	0	0	0
DR	95	93	8	3	0	0	0
RTcal	88	5	0	0	0	0	0
Command	89	88	0	0	0	0	0
TRext	8	0	0	0	0	0	0
M	99	95	88	5	0	0	0
CRC5	99	95	95	2	0	0	0
CRC16	94	93	92	0	0	0	0
Slot	99	99	99	86	89	93	94
Q_query	94	91	93	94	89	94	63
session	94	94	46	0	0	0	0
target	96	95	48	38	33	0	0

In this section we summarize the effect of SEU faults for every registers in the tag after extensive fault injection campaigns performed for the three workloads. In all experiments a single fault has been injected. The last results give signification information about the EPC C1 Gen2 tag behavior in the present of SEU faults.

4.5.5.4 *TRcal*

We see that the number of detected errors when fault are injected in this register TRcal is very important. This justifies the importance of this parameter. This parameter is used to calculate the BLF (Backscatter low frequency), which is the clock frequency of all the modules that handle the response of the tag as FM0 and Miller encoders.

$$BLF = \frac{DR}{TRCAL} \quad \text{Equation 4.1}$$

With a faulty clock, symbol encoding will be correct but this involves a de-synchronization between the tag and reader (Figure 4.40).

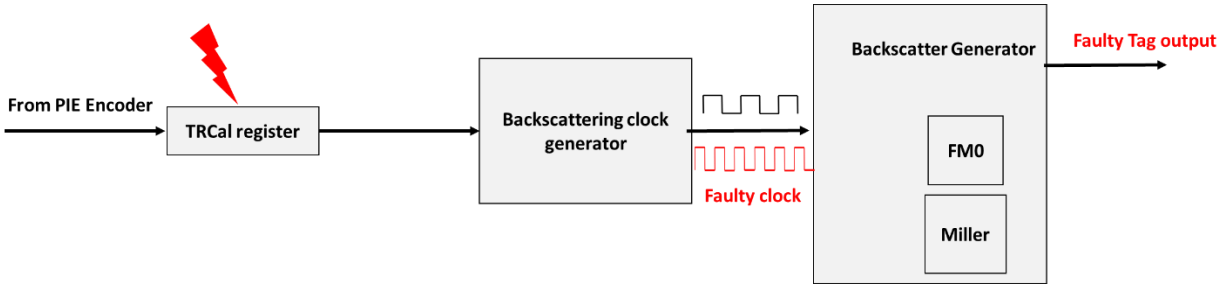


Figure 4.40: TRcal Registrer SEU faut effets

4.5.5.5 *Divide ratio (DR)*

This register is also used to calculate the BLF (Backscatter low frequency), fault injection in this register induces the same effect as RTcal which explains the convergence in the detected fault number with RTcal.

4.5.5.6 *Reader to tag calibration RTcal*

The value of this register is used as the time reference to decode the symbol. The reader fixes this parameter at the beginning of each inventory round. Then, the reader sends this parameter periodically during inventory round at each frame. This last point explains the low rate of detected errors on this parameter. Indeed, injected faults do not affect the tag response when a parameter update is done before using it. As shown in Figure 4.41 the only effect introduced by the fault injection in this RTcal register affects the response time T1 between the command received and the response sent by the tag (the tag calculates the T1 time using RTcal).

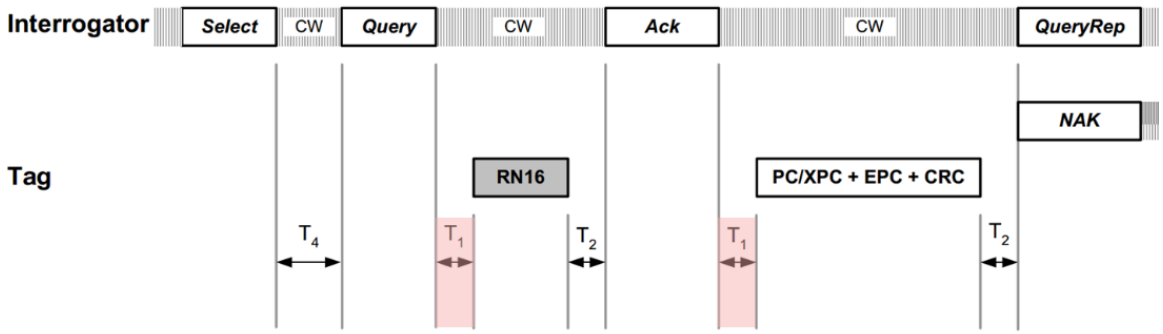


Figure 4.41: Impact of RTcal on the tag to reader communication timing

The tag has been designed to protect this parameter against SEU during inventory round. We detail in the following the protection mechanism. RTcal is sampled N times during each inventory round. Then we compare the sampled values to choose the most repeated value and to use it as a fixed RTcal during all inventory rounds.

4.5.5.7 *Command register*

The command register is a very important register. It is the parameter that is used by the state machine to control the entire tag. In our tag architecture, the command register is rewritten at each clock cycle. This explains the results of workload 1, for which the number of latent error is more than the detected error. The results of workload 3 confirms that if the duration of fault exceeds the required time to rewrite the command register, this will generates a catastrophic error and the tag will remain without response.

4.5.5.8 *Target and M parameters*

The tag captures these two parameters target and M when it receives a Query command from the reader. The fault effects on these two parameters when the injection is done during a query command or when the finite state machine accesses these two parameters is described in Figure 4.42.

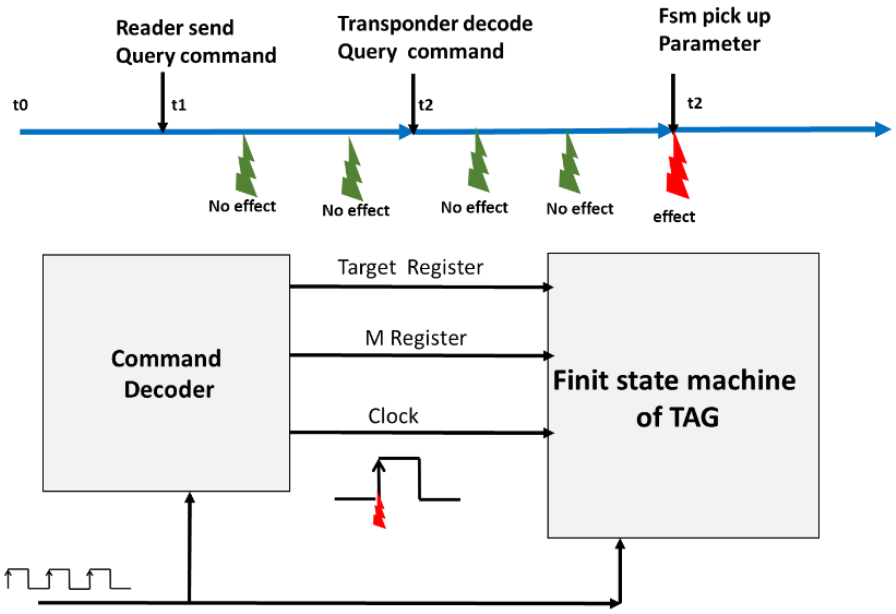


Figure 4.42: M and target register fault injection effect

4.5.5.9 *The slot counter and Q parameter*

The Slot counter controls when the tag may answer using up down counter with preload value. The preload value is generated using PRNG and the Q value. This value is between 0 and 2Q.

If the fault is injected in the slot counter or in Q, then the duration before the tag answer will be modified. Thus, the tag responds before or after its allowed slot. But the tag will answer. As shown in Figure 4.36 there is a very high rate of latent error when the fault is injected in slot counter or Q register.

4.5.5.10 *The session parameter*

This register is used to manage tag population and helps the reader to select which tag can participate in current inventory round. A flag of 2 bits is used by the tag to know if it is concerned by this inventory round or not. If we inject fault in this register the finite state machine stops in an arbitrate state and compares every time receiving a query command if the flag stored in the tag is the same as the one send by the reader to begin an inventory round. As we see it is a very sensitive register which explains the high number of detected error during the fault injection campaign.

4.5.5.11 *Fault propagation path*

The workload 2 allows us to trace the fault propagation paths inside the tag for every fault injection campaign. Figure 4.43 shows an example of fault propagation when we inject fault in TRcal register. This fault involves a faulty clock which will be used for the backscattering. Then the FM0 or Miller modules generate faulty rate output that the reader will not read correctly.

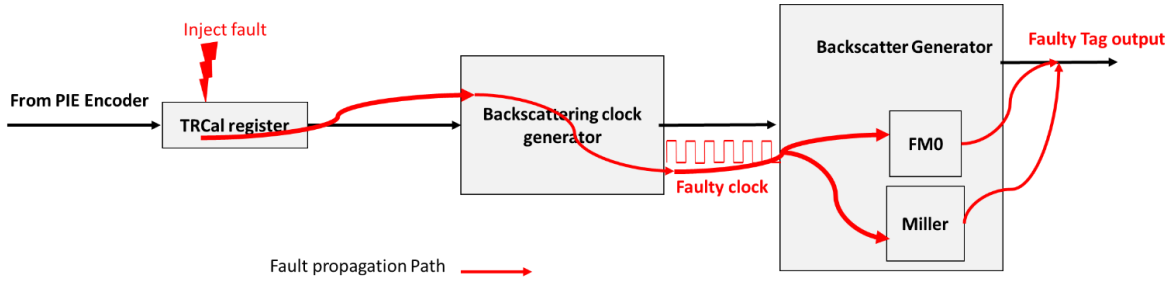


Figure 4.43: Fault propagation in the case of TRcal register

4.6 EPC C1 Gen2 tag enhancement reliability

4.6.1 *Countermeasure against SEU error to enhance the tag read rate*

Previous experimental results allow us to identify the most sensitive data: 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 only 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 4.44, the TMR [Lyon1962] technique consists on the triplication of the target component to protect.

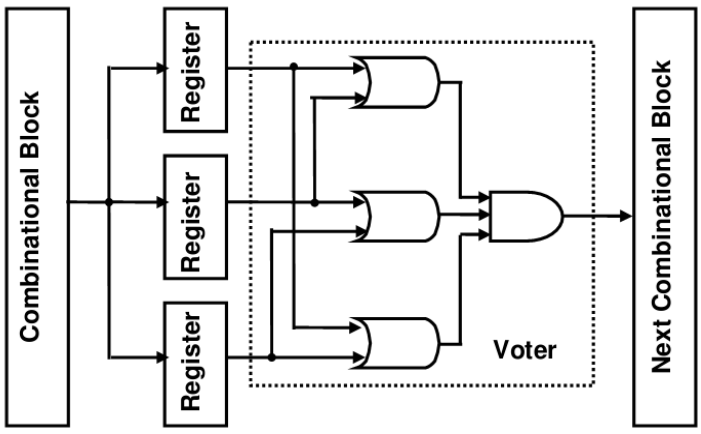


Figure 4.44: TMR Protection

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. Figure 4.45 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.

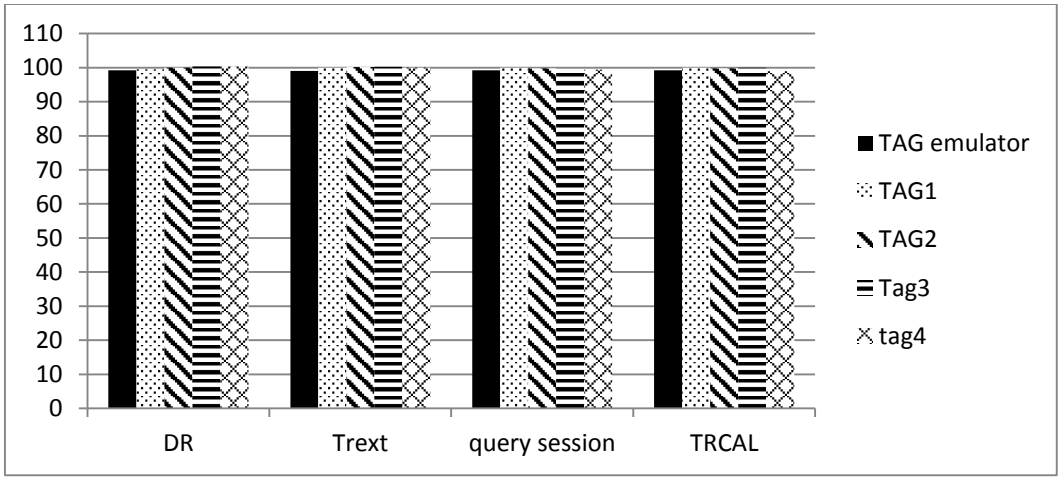


Figure 4.45: Fault Effect with Redundancy

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 this proves that TMR efficiently protects the sensitive registers. The added surface for embedding this TMR corresponds to 30 flip-flops (FF): triplication of the RTcal adds 18 FF plus 3 FF for the DR register, 3 FF for the TRext 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 4. 8 shows the results reported by the Xilinx ISE tool after the place and route.

Table 4:8: Device utilization table for the tag with TMR and tag without TMR

	Tag Without TMR	**Tag using TMR**
# Slice Registers	382	382
# Slice LUTs	379	380
# LUT-FF pairs	782	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.

4.6.2 *Countermeasures against the SEU faults*

To enhance the level of dependability of RFID tag that will be used in critical systems or in harsh environments, mechanisms for protection against SEU errors must be used.

A robust mechanism must ensure fault detection in a time less than error latency to avoid system failure. For this, mechanisms to enhance the reliability of RFID tag must be lightweight in terms of area and energy. Before detailing our contribution we will give a brief overview of 2 fault detection and system recovery techniques: sequence state integrity testing and spatial redundancy

4.6.2.1 *Sequence states integrity testing*

The first class of detection is to verify that the state sequence is performed correctly. For this purpose, during the circuit design, the allowed sequences are determined.

The allowed state sequences are {ready → arbitrate; arbitrate → reply; reply → Akn; ; Akn → open ; Akn → Arbitrate; reply → arbitrate}. When running the tag, we need to ensure that every effective transition of the FSM is included in the previous authorized transitions.

4.6.2.2 *Spatial redundancy*

Spatial redundancy uses bits to encode the data to be protected. This spatial redundancy can be implemented using Error Corrector Code (ECC). The faults detected by these ECCs are limited as shown in Table 4.9. For example, parity code can detect only one bit error, and Berger code deals only unidirectional errors (eg. transitions 0 to 1 or 1 to 0).

Table 4:9: ECC types using with prediction circuit

Code type	**Properties**
Parity check	Detect only one bit error
Berger code	Detect all unidirectional error
Bose-line code	Detect unidirectional error depending on the used bits number
Dong code	Detect unidirectional error depending on the used bits number

To protect calculations using ECCs, a solution called “prediction code " has been proposed. Fig. 4.46 shows an implementation of this solution. In this figure, A and B are two

inputs of the combinational circuit F. S = F (A, B) is the output of this circuit. Ca, Cb, and Cs are respectively the ECCs of A, B, and S. The prediction circuit P is coupled with the original circuit F for generating another output ECC called Cs'. The prediction equation P is derived from the mathematical properties of the ECC. By comparing Cs and Cs', we can validate the output S.

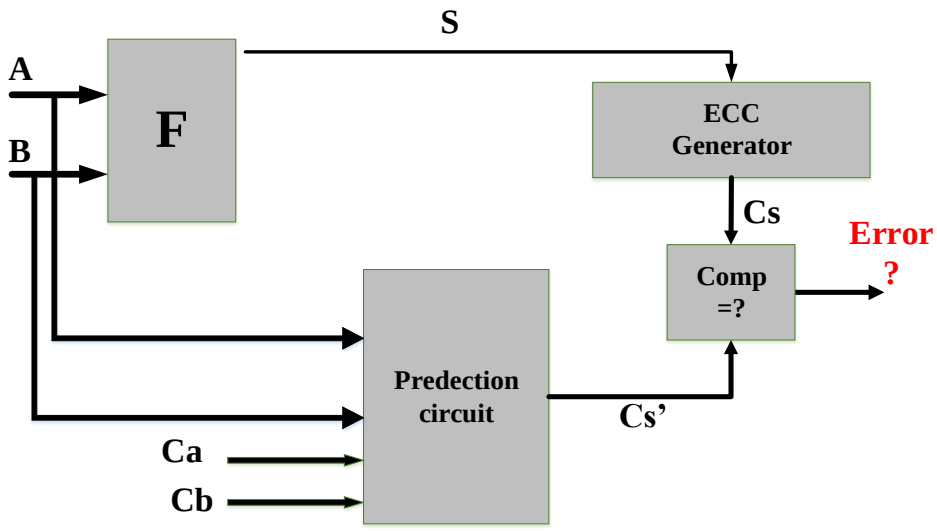


Figure 4.46: Design of hardware redundancy using prediction

Since the tags are low cost and passive, they support only simple operations. Most of the time and spatial redundancy techniques are too costly in term of area and power consumption. In this work we will present two lightweight methods to improve the reliability of digital RFID tag based on two different approaches. The first using sequence states integrity testing using assertion based on-line fault detection and the second using spatial technique is based on a simple ECC (cross parity check technique) coupled with parity prediction.

4.6.3 *Assertion Based On-line Fault Detection*

In this approach we propose a new RFID tag monitoring technique based on adding an infrastructure circuit which simultaneously monitors and saves faulty tag behavior to enable advanced RFID diagnosis functions, and mainly reinforce tag security against fault attacks. The added infrastructure circuit is essentially constituted of hardware assertions that permit on-line fault detection in the tag. Saved information about detected faults on the tag can be then read using an RFID reader.

Complexity of new systems involved the emergence of new verification methods for full system debugging. For this purpose, Assertion Based Verification (ABV) has been introduced

and becomes nowadays preferable to designers to validate their products. An assertion refers to a safety property that must hold globally in a system. If an assertion is ever violated, this indicates that a bug is present [TAO2009].

Assertion languages such as PSL (Property Specification Language) and SVA (System Verilog Assertion subset) appeared with the aim of satisfying debugging need at a high level. Moreover, making efficient assertions at RTL level (Register Transfer Level) throughout these languages and following formal methods, simulation and emulation allows yielding to an important bug coverage ratio and leading to more accurate systems [Boule2005].Assertions are introduced in a design with the aid of dedicated libraries which contain a set of assertion checkers that verify specific properties of a design. Several ABV libraries are offered by numerous companies. Among these libraries, the standard Open Verification Library (OVL) developed by Accellera Inc. provides designers vendor-independent interface [Accellera2013].

Although assertion is used for design verification, several researches suggest the use of assertion for post-silicon purpose such as post-fabrication silicon debug, on-line testing and fault-tolerant systems design [Boume2007][Riazati2006][Kubalik2006].When assertions cannot be directly implemented in hardware because they are written in a higher-level language and dedicated to be monitored by simulation, multiple approaches are proposed to synthesize assertions given in PSL or SVA languages and thereafter permit the hardware implementation [Boume2007] [Riazati2006]. Furthermore, there is many checker generators that permit efficient generation and implementation of hardware checkers [Boule2005][Abarbanel2000].

In this field, we have used assertions for on-line fault detection in RFID tag. Standard OVL checkers have been selected to build these assertions with a benefit of portability and effectiveness of OVL as well demonstrated in [Kakoee2008].

Making multiple and efficient hardware OVL assertions in a circuit allow to monitor its activity. When the circuit works properly, no violation is produced neither detected. In case a violation is detected by an assertion checker, its fire signal is set indicating the presence of this violation which signifies that one or multiple faults are produced inside the circuit.

The added monitoring circuit includes thirteen assertions checkers distributed all over the tag as shown in Figure 4.47 and introduced in Table 4.10. Among these checkers, eleven ones are OVL checkers used to control the main tag signals of synchronous frame reception, synchronous frame emission and synchronous tag state transitions. The used OVL checkers

types have been selected in such a way to monitor the whole tag behavior: for example, the checker c9 “ovl_one_hot” check that a frame reception phase does not match frame proceeding phase neither frame sending phase.

Nevertheless, as the synthesizable checkers of OVL library do not enclose the total reliability aspects that we aim to cover in the RFID tag, we developed two other checkers (c12) and (c13) in order to monitor transitions of tag FSM and PIE decoder states machine.

As depicted before, the monitoring circuit provides the numbers of faults detected and reset signal. Therefore, two counter registers, FSM fault counter and OVL fault counter, are inserted to count detected faults by each type of checkers Figure 4.47. These registers are mapped at the top User memory address to permit reading and erasing their contents. Reset signal is generated after fault detection by any implemented checker, this permit to avoid any unwanted tag task.

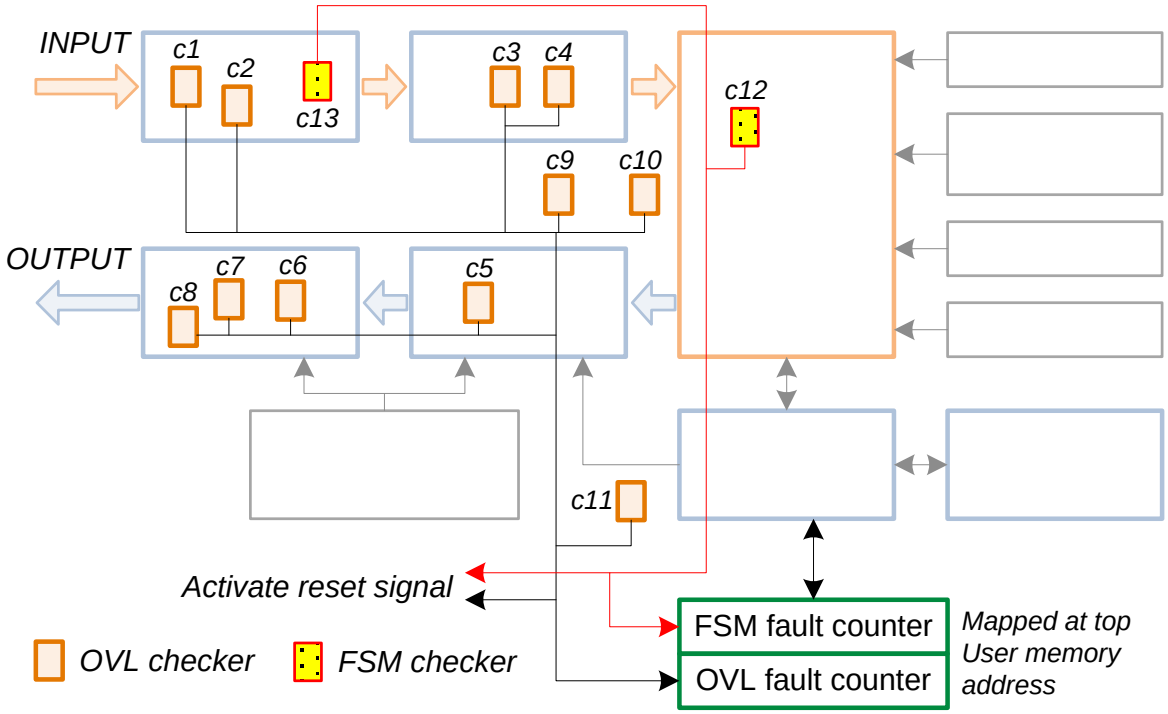


Figure 4.47: Proposed on-line fault detection approach based on hardware assertions and applied on a developed RFID tag architecture.

Table 4:10: Type of used assertion

Assertion checker type	Assertion	Assertion name (Figure 5.54)
ovl_always	3	c2, c4 and c5
ovl_implication	4	c6, c7, c8 and c10
ovl_next	1	c1
ovl_one_hot	1	c9
ovl_zero_one_hot	2	c3 and c11
FSM checker	2	c12 and c13

Monitoring circuit evaluation was realized by numerous experiences whence 2051 faults totally are injected. An experience starts by launching tag scan process by RFID reader during several seconds. The fault injection with a rate of 0.006% is automatically activated during communication while tag is running. During fault injection, only one fault is injected randomly in one signal between 24 selected signals by inversing its value. After scan process is stopped, fault injection is automatically halted and then results are harvesting through two ways:

The number of faults detected by assertion checkers is obtained through RFID reader by reading data stored at the top address (07h) of tag User memory as shown in Figure 4.48.

The number of total injected faults and the number of effective detected faults are obtained through 8 LEDs of FPGA board with the aid of on-board switches by multiplexing internal data.

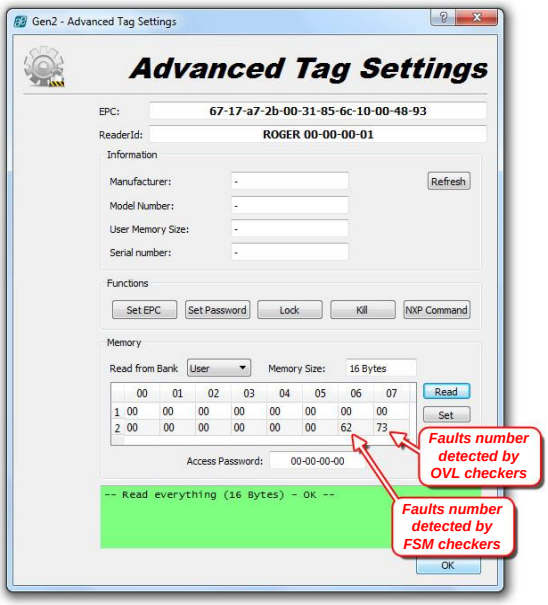


Figure 4.48: Reading detected faults numbers with RFID reader software.

The obtained results are given by Table 4.11, we see that total number of faults detected by implemented checkers is 3358, knowing that one fault injected can be produce multitude faults and thereafter multitude fault detections about checkers. Wherefore, we deliver in this table the effective number of faults detected by these checkers which is the turn of 73%.

Table 4:11: Fault detection effectiveness

Assertion type	**Injected faults**	**Detected faults**	**Effective detected faults**	
			Number	***Percentage***
FSM checkers	2051	1542	1011	49.29%
OVL checkers		1816	492	23.98%
Total		**3358**	**1503**	**73.28%**

The main goals targeted by integrating this feature to RFID tag is to allow an enhanced diagnosis within RFID systems as well protecting tag against fault attacks. Added infrastructure is constituted essentially by several hardware developed and OVL checkers. The fault detection rate is saved in tag internal memory and can be read or reset through RFID reader.

The area occupation of the developed tag before and after addition of on-line fault detection infrastructure circuit is illustrated in Table 4.12; the obtained results are based on Xilinx Spartan-3E XC3S500E FPGA implementation and demonstrate that Slice occupation is increasing by 17.60% after adding the infrastructure circuit. Noting that slice occupation of the tag before and after adding the proposed circuit is 13% and 15% respectively.

Table 4:12: RFID tag area occupation on Spartan-3E FPGA before and after implementation of on-line fault detection infrastructure circuit

Circuit	**Area occupation**	
	Slices	***Flip Flops***
Developed RFID tag	625	373
Added infrastructure circuit	110	46
Tag with infrastructure circuit	735	419
Area increase	**17.60%**	**12.33%**

The proposed approach has been implemented and tested using developed tag architecture based on finite state machine and compatible with EPC C1 Gen2UHF RFID standard tag

emulation platform with physical random fault injection to evaluate fault detection capability of proposed approach. Experimental results have shown that 73.28% of the injected faults have been detected.

4.6.4 *Error code correction for Fault Detection*

The second approach increases the robustness of RFID tag using error code detection. Several solutions are possible such as CRC cyclic codes or block codes, Hamming distance code, etc. Some of these techniques can only detect errors and others detect and correct errors at the same time. Since these techniques are costly in terms of occupied area and energy used for computing we have used the parity checking techniques.

The parity check (sometimes called VRC for Vertical Redundancy Check or Vertical Redundancy Checking) is one of the simplest systems for error detection control. In this technique an additional bit (called parity bit) is added to a number of data bits called code word (7 bits generally, to form a byte with the parity bit). The value of this additional bit (0 or 1) is such that the total number of bits at 1 is even. Parity checks detect only an odd fault number, so it can detect only 50% of errors. This is its major drawback.

Cross-parity prediction (CPP) technique proposed in [Pfanz2002] is used for register on-line checking. Figure 4.49 shows an example of register file for 8 x 8 bits register with multiple faults in register 0 and 2 which can be detected by parity check, however faults in register 1and 4 cannot be detected.

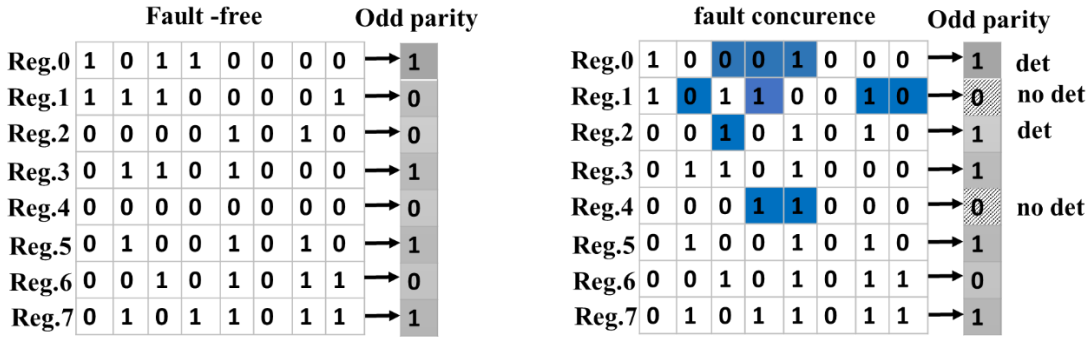


Figure 4.49: Limitation of simple parity check [Pfanz2002]

The proposed solution in [Pfanz2002] calculates the parities of row and column and stores it in two registers to use it during checking. In Figure 4.50 the double error detection (example as shown in Figure 4.50 and exactly in register 4) becomes possible by checking both the row and column parities.

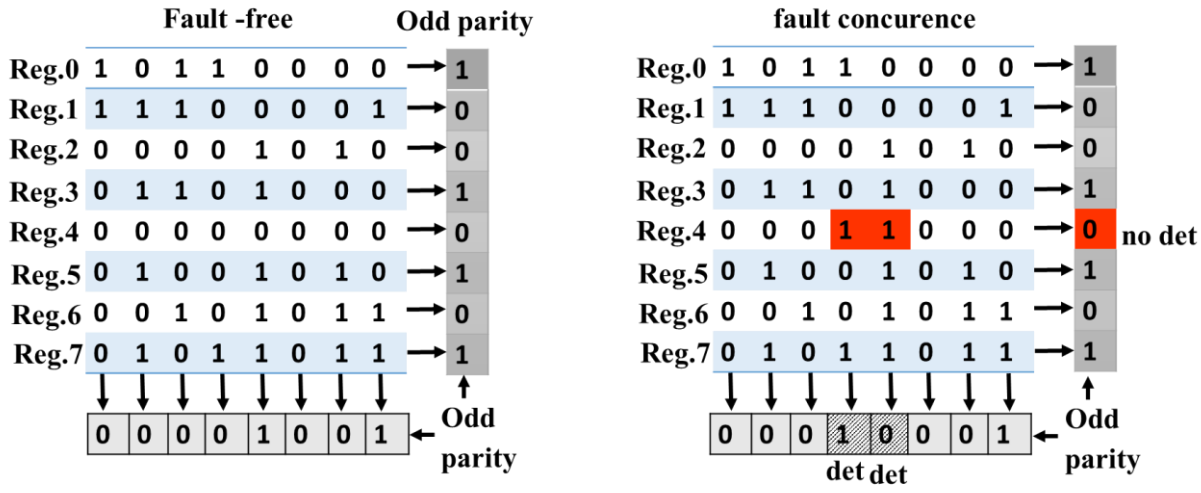


Figure 4.50: Row-and column-Parity checks [Pfanz2002]

But the last technique cannot detect all the faults. For example, the faults in registers 4 and 5 are not detected as illustrated in Figure 4.51. To detect faults in this case, the authors of [Pfanz2002] introduced the cross parity check.

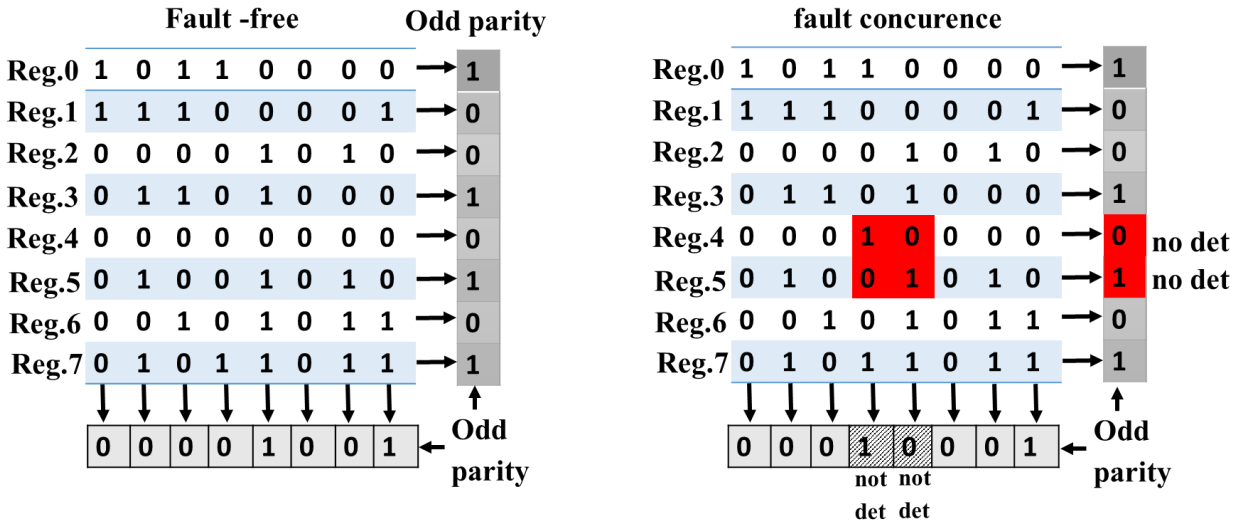


Figure 4.51: Limitation of Row and column-parity check [Pfanz2002]

The cross parity check technique calculates the diagonal–parity from the most significant bit of the most significant register to the least significant bit of the least significant register (see Figure 4.52). If the size of the register we want to calculate the parity is Y and the number of register is X, the row parity vector r [(X-1): 0], the column parity C[(Y-1): 0] vector and the diagonal vector d[max(X, Y) -1 : 0] form the cross parity.

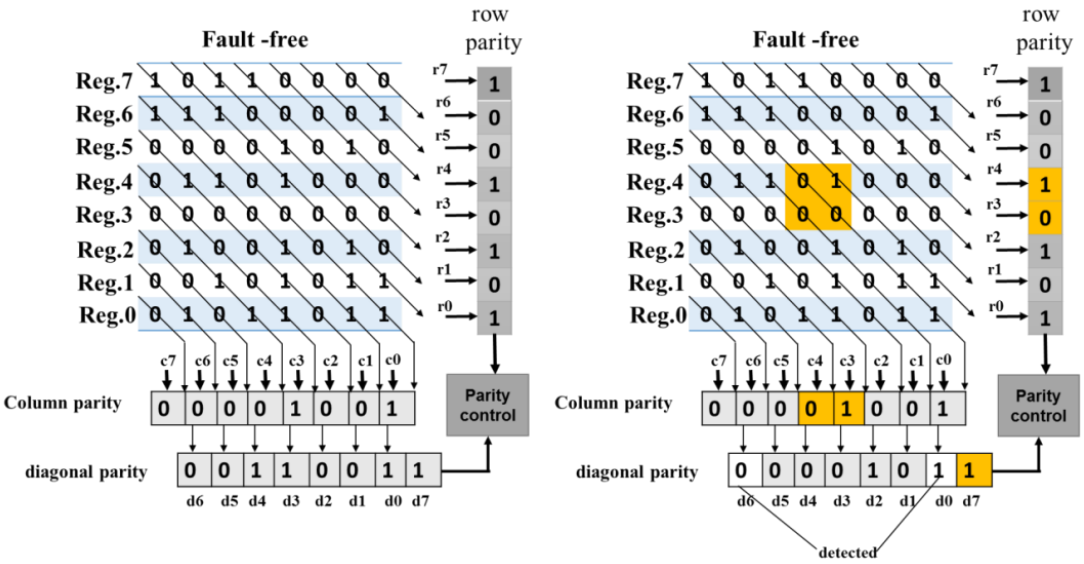


Figure 4.52: Cross-parity organisation for register files [Pfanz2002]

The diagonal parity can be calculated from the most significant bit of the most significant register to the least significant bit of the least significant register.

$$R_X = M_{X,Y-1} \oplus M_{X,Y-2} \oplus M_{X,Y-3} \oplus \ldots \oplus M_{X,0}$$

$$C_Y = M_{X-1,Y} \oplus M_{X-2,Y} \oplus M_{X-3,Y} \oplus \ldots \oplus M_{0,Y}$$

Eg: The Crosse parity of 5x5-bit register file can be calculate as flowing:

$$D_{Z=4} = M_{4,4} \oplus M_{3,3} \oplus M_{2,2} \oplus M_{1,1} \oplus M_{0,0}$$

$$D_{Z=3} = M_{4,3} \oplus M_{3,2} \oplus M_{2,1} \oplus M_{1,0} \oplus M_{0,4}$$

$$D_{Z=2} = M_{4,2} \oplus M_{3,1} \oplus M_{2,0} \oplus M_{1,4} \oplus M_{0,3}$$

$$D_{Z=1} = M_{4,1} \oplus M_{3,0} \oplus M_{2,4} \oplus M_{1,3} \oplus M_{0,2}$$

$$D_{Z=0} = M_{4,0} \oplus M_{3,4} \oplus M_{2,3} \oplus M_{1,2} \oplus M_{0,1}$$

As shown in the Figure 4.52, the cross-parity check uses the three parities which cover all errors.

This technique has been implemented as described in the Figure 4.53 using the spatial redundancy technique previously described in section 4.6.2.2. This implementation has been tested with several registers of the tag. It allows us to perform on-line monitoring of all vulnerable registers of the tag.

The cross parity register will be permanently compared with the cross parity prediction (CPP). The CPP will be updated each time the tag changes the parameter values of one of these monitored registers.

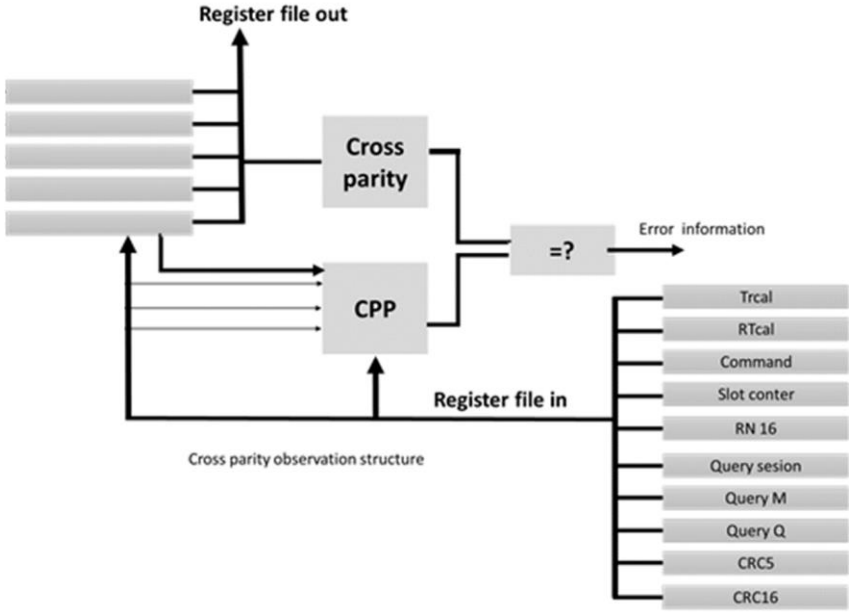


Figure 4.53: Cross parity coupled with the prediction parity architecture [Pfanz2002]

The proposed approach has been implemented using the weakest tag registers (these registers have been previously identified thanks to fault injection). Then we have evaluated the fault detection capability of the proposed approach. Experimental results have shown that 100% of the injected physical random faults have been detected. Thus the cross parity check coupled to parity detection showed a great potential to detect and localize faults.

This implementation has also been enhanced by adding a correction mechanism based on a simple technique allowing rewriting the detected faulty bit in the faulty register.

The area occupation of the developed tag before and after the addition of this on-line fault detection using cross parity check is given in Table 4.13. The obtained results demonstrate that Slice occupation is increasing by 12.48 % after adding the cross parity check circuit and the parity prediction.

Table 4:13: RFID tag area occupation before and after implementation of cross parity checks

Circuit	**Area occupation**	
	Slices	***Flip Flops***
Original RFID tag	625	373
Cross parity and CPP circuit	78	32
Tag with Cross parity and CPP circuit	703	405
Area overhead	**+12.48%**	**+8.57%**

4.7 Conclusion

In this chapter we have presented the RFIM platform. RFIM helps us to study the robustness of UHF EPC C1 Gen2 tag in the presence of faults and in real environments. This study identifies the most sensitive registers of a tag and parameters of the EPC C1 Gen2 protocol. This later helps us to develop a low cost and more robust and secured tag architecture by the proposal of different solutions and countermeasures.

RFIM permits to validate the new tag architecture in real environments. In addition, RFIM allows observing all the faults effects in the tag during the RFID system operations (with one or several different tags and readers). Using RFIM, it's possible to experimentally validate secure architectures and to analyze the faulty behavior of complex RFID systems by studying the propagation of a fault after its injection in a specific tag. The RFIM platform allows emulating any types of fault (transient or permanent stuck-at faults) in real RFID environments.

Chapter 5

5 Threat evaluation of Hardware Trojan within RFID System

5.1 Introduction

Hardware Trojans are malicious hardware components embedded within the chip by an adversary, in order to disable, destroy a system, or leak confidential information at some future time. The Hardware Trojans can be inserted in any step of IC manufacturing. Such malicious circuitries could be a great threat for the governments, commercial, financial or military institutions and everywhere where RFID technologies are used.

Many studies have been done concerning Hardware Trojan insertion and Hardware Trojan detection within Integrated circuits [Mark2011] [Karri2010] [Chakraborty2008] [WOLF 2008]. These studies show that Hardware Trojan is a real threat to consider when designing critical systems. RFID systems integrators are using many different commercial tags and sometimes have to deal with tags they do not have chosen. In this context, how integrators can guarantee that the tags that are being used are safe and do not embed malicious hardware which can jeopardize the system safety or security? To detect Hardware Trojans, companies need to verify the proper operation of their products in different ways, which of course can have a great time and financial costs. The earlier discussed complexity of RFID systems makes both the evaluation and the protection against the HT threat very complex. The aim of this chapter is to show that the RFIM platform can be a powerful tool to evaluate the HT threat within RFID system and can be used to develop multi-level adequate countermeasures against this threat.

The first part of this chapter focuses on Hardware Trojan by presenting general research results on HT insertion and detection. Then the second part of this chapter shows how the hardware emulation platform *RFIM* Emulator helps to evaluate some trigger and payload of Hardware Trojan, in order to evaluate associate risks on RFID EPC C1 Gen2. Finally, we discuss countermeasures which could be designed at the system level in order to protect RFID systems against untrusted RFID tags.

5.2 Hardware Trojans

5.2.1 *Hardware Trojan Components*

As shown in Figure 5.1, a Hardware Trojan (HT) is composed of two parts: the trigger and the payload. While the trigger part acts as a sensing circuit which is used to activate the HT, the payload is the malicious activity of the HT. Most of the time Hardware Trojans remain in dormant state and are activated by the trigger logic. This one can for instance be a timer, or the occurrence of a rare event. In the specific case of RFID systems, the HT could be trigged by a dedicated action of the RFID tag or specified commands coming from the Reader. The payload is the malicious operation of the HT as shown in Figure 5.1. This operation takes the original and correct signal S from the circuit and modifies it to a malicious signal S' when the activation condition of the trigger logic is true. The trigger inputs can be internal signals of the IC, or external signals coming from outside of the IC. The aim of this modification is creating a backdoor to attacks the IC. These attacks can be denial of service operation, deactivating the RFID tag (by activating the kill command), leaking the password of RFID tag memory or changing the behavior of RFID tags.

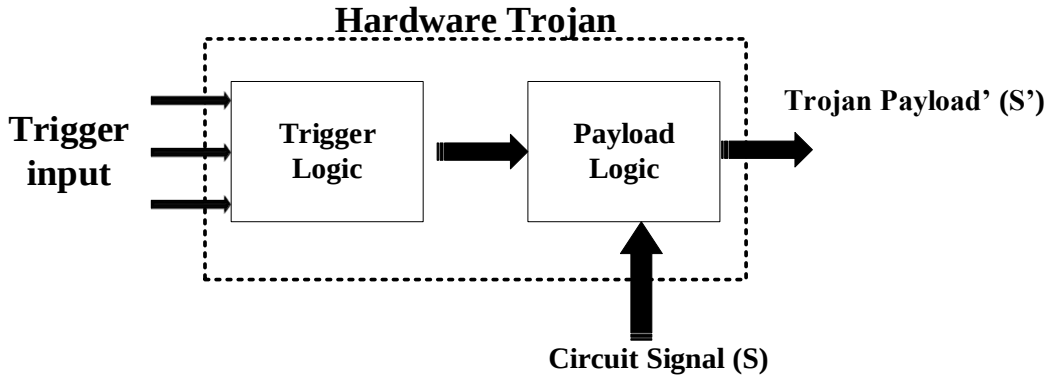


Figure 5.1: General structure of a Hardware Trojan in a design

The small size and the ease of Hardware Trojan concept make it a serious threat in modern circuits.

5.2.2 *Hardware Trojan Insertion phase and location*

As shown in Figure 5.2 several parties are involved in the IC design life cycle:

Foundries: they are the semiconductor manufacturers. They take the hardware design in GDSII industrial format to fabricate the ICs.

IC designers: designers of the IC who have generated the GDSII.

IP vendors: they develop and sell intellectual property cores to be used by the IC designer for integration in final chip (e.g. RFID tag core, ASK modulator IP).

EDA tool vendors: they provide EDA tools for IC designers and IP vendors, to facilitate the design of large scale integrated circuits.

IC end users: companies or individuals who purchase the designed chips from IC designers.

The insertion of Hardware Trojan can be done in any stage during IC manufacturing processes, beginning by specification, design, and up to verification and manufacturing.

.

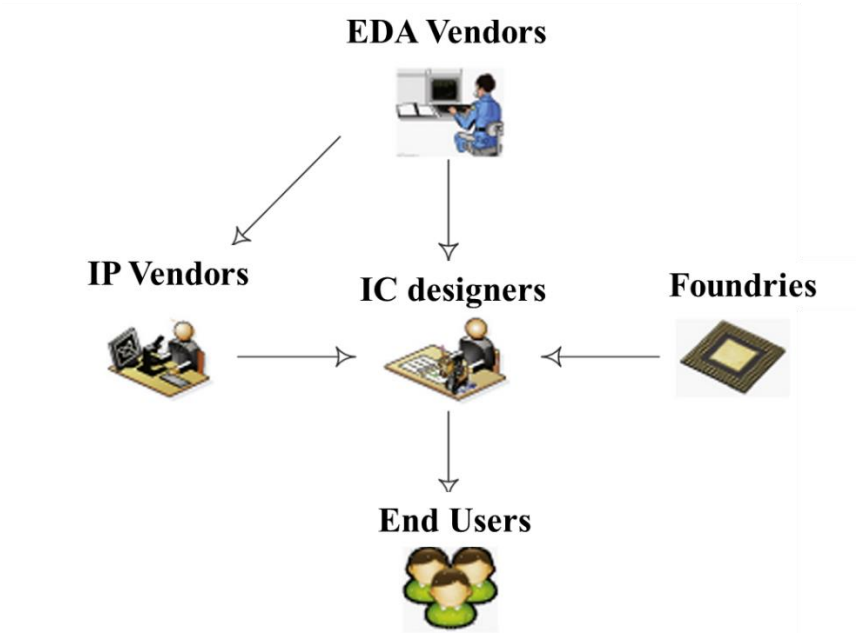


Figure 5.2: IC life cycle [HeLi2016]

Due to global economic pressures, manufacturing companies have spread around the world in order to minimize the manufacturing costs. Circuits could then be modified by an adversary within these foundries to add malicious circuitries [Jin2009]. We note that malicious circuitry can also be added during the design phase. It can be done either intentionally by the IC designer or by a malicious designer or subcontractors (IP provider, design tools …) [Chakraborty2009], to gain an advantage on the future use of the circuit.

Increasing outsourcing for the design, and the number of entities and person involved in the IC life cycle increases the possibility, and gives more opportunity for Hardware Trojan insertion in the final Hardware Figure 5.3.

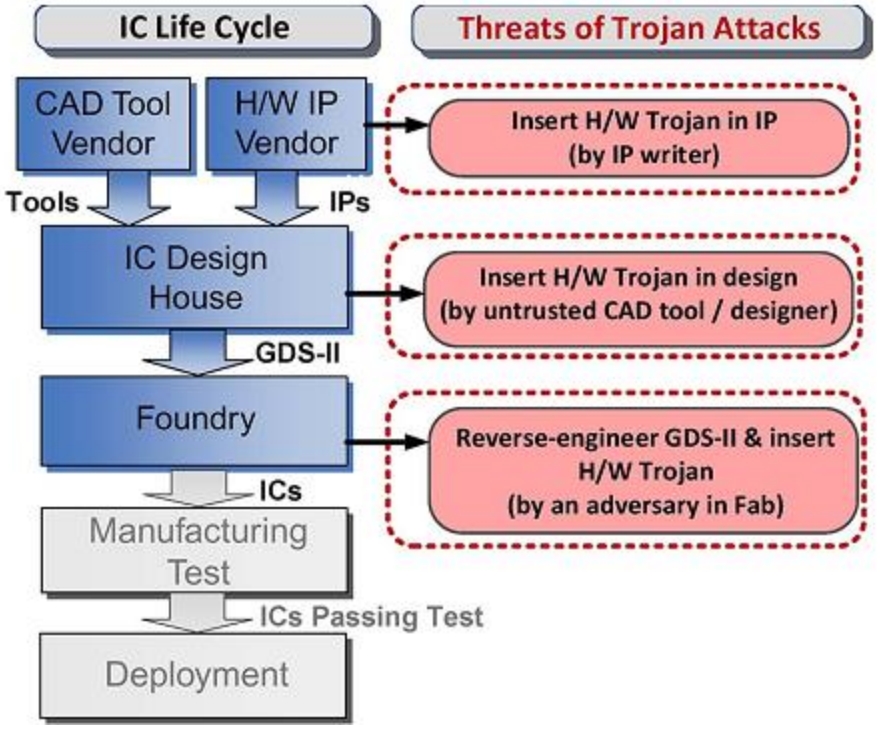


Figure 5.3: Hardware Trojan attacks by different parties at different stages of IC life cycle [Bhnia2014].

The foundries can insert Hardware Trojan in chip by modifying the dopant level as shown in work [Shiyanovskii2010], or in mask layout [Becker2013].

The IC designer uses the acquired IP from IP vendor. This last can have included a malicious function [Lin2008] [King 2008]. The IP vendor can insert the Hardware Trojan at RTL level by adding a malicious code, or by modifying the synthesis and compilation macro to insert HT during design synthesis or during the Place&Route step.

EDA tools are used in all critical design stages. This allows the EDA vendor to put a malicious code which can leak information about the designed IC [Qu2014].

5.2.3 *Trojan Taxonomy*

As discussed above, Hardware Trojan can be very different from each other, depending on the activation type, the effect or the way it has been inserted. It is then necessary to have a classification according these different attributes before to evaluate the threat and to design

countermeasures. Different Hardware Trojans classifications have been proposed, in Figure 5.4 we show a global overview of Hardware Trojan classification combining the different classifications that have been proposed in the literature.

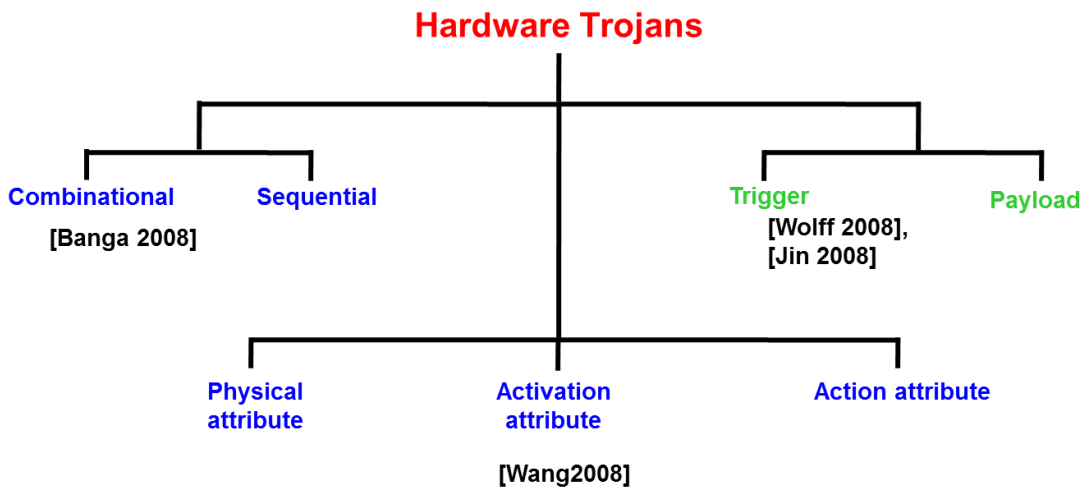


Figure 5.4: Global view of Hardware Trojan Taxonomy

The authors of [Chakraborty2009] and [Bhunia2010] proposed a classification shown in Figure 5.5, extended from [Bhunia2008] that is based upon trigger and payload mechanisms.

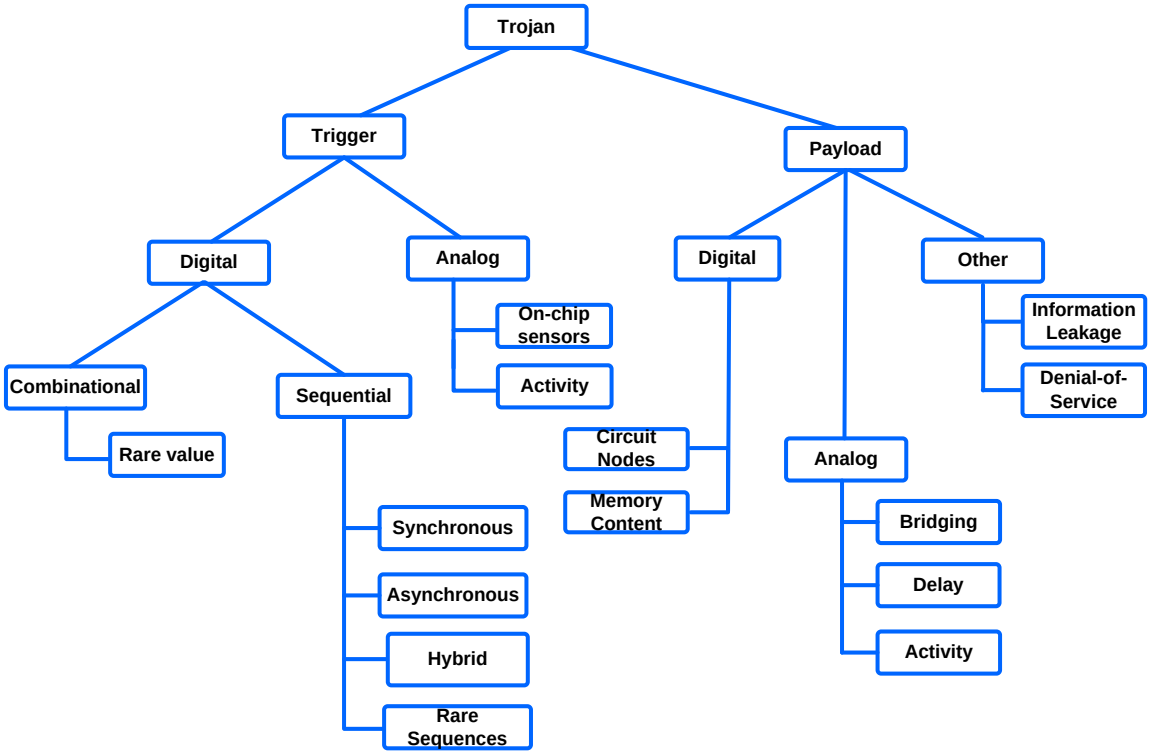


Figure 5.5: Hardware Trojan Taxonomy: [Bhunia2010]

5.2.4 *Hardware Trojan trigger Taxonomy:*

As we have seen there are different methods for Hardware Trojan insertion, and each method has an impact on the IC. As we know Hardware Trojan remains dormant until the trigger condition is true to run the malicious activities, in the following we present some works about the various Hardware Trojan trigger types as shown in Figure 5.5.

5.2.4.1 *Analog trigger:*

In [Chen2008] analog HT trigger mechanism has been proposed. It uses ring oscillators to add more activity to a circuit. This generates more heat, then the temperature sensor is used to detect the rise in temperature and trig the malicious operation.

5.2.4.2 *Digital trigger: combinational and sequential*

a) Combinational

The Hardware Trojan is activated when specific values are detected simultaneously at specific internal circuit nodes, as shown in Figure 5.6 The required conditions to trig the payload in node C is A=1, B=0. Under this condition, the correct output C will be replaced by the faulty output C'. We can note here that in a way, the same principle that we have used for fault injection by replacing the right signal by a faulty signal is used. This trigger type is possible in the case of passive RFID, because it does not require a lot of space (less than 10 gates), and its power consumption is acceptable.

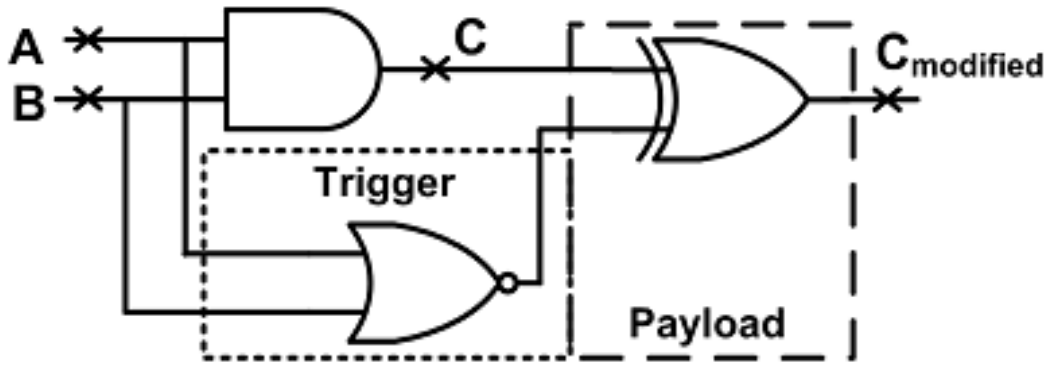


Figure 5.6: Combinational triggered HT [Chen2008]

b) Sequential:

Sequentially triggered HTs are activated by the occurrence of a sequence of operations.

The simplest sequential trigger is a simple synchronous counter which trigs a malfunction when reaching a particular count as depicted in Figure 5.7. Then the correct signal ER is modified to the incorrect signal ER*.

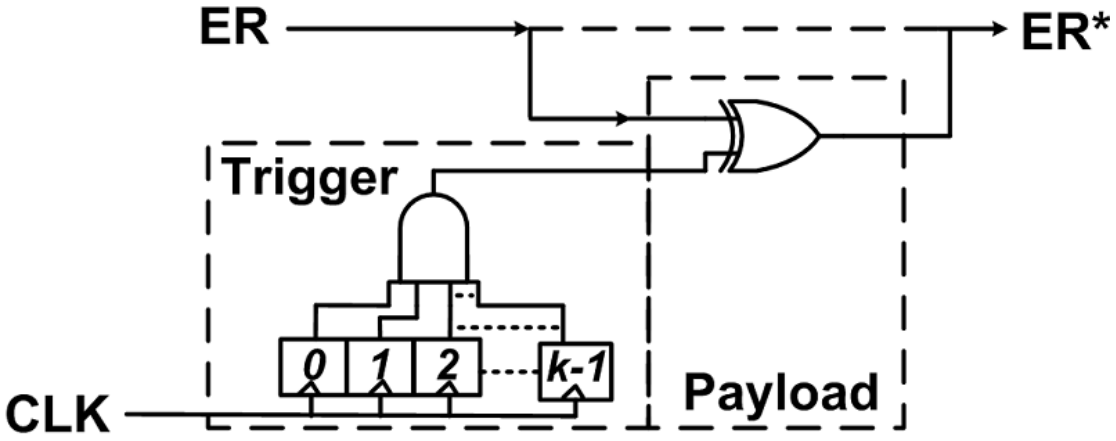


Figure 5.7: Synchronous counter Trojan [Bhunia2010]

- ***Rare sequence Hardware Trojan trigger***

In this case the trigger condition is based on a sequence of rare events. The event can be issued from state machine states. This type of trigger is more complex to detect during IC testing and verification stage, because it requires satisfying a sequence of rare conditions in the internal circuit nodes to be activated. In the second part of this chapter we will show with an example the effects of this trigger on a RFID system.

The sequential trigger type is possible in the case of a passive RFID tag, because it does not need a lot logic gates.

In Table 5.1 we resume and compare the proposed Hardware Trojan trigger for area, energy consumption, insertion difficulty, discretion and the applicability for RFID system.

Table 5.1: Hardware Trojan Trigger

Type	Area	Energy consumption	Insertion difficulty	Discretion	Applicability In RFID
Analog trigger	++	++++	high	+	Not applicable
Combinational	+	++	lower	++++	Applicable
Sequential	++	+++	medium	+++	Applicable

The analog and sequential triggers require more logic gates compared to combinational triggers. These combinational triggers can be implemented using only a few logic gates. More gates also involve more power consumption. The analog and sequential triggers require more

sophisticated operations (their insertion is more difficult than the one of the combinational trigger). In addition, the analog triggers can be detected using side channel methods (Temperature or power measurements).

We conclude that the analog trigger is more difficult to apply in the context of low cost and low power UHF RFID tags. Indeed the UHF tags can't ensure a sufficient energy for oscillators and sensors at the same time. The insertion of combinatorial and sequential triggers is easier, since the tag already contains combinational and sequential logics, which makes difficult to differentiate logic trigger from the logic of the RFID tag.

5.2.5 *Hardware Trojan payload Taxonomy*

The payload is the threat action associated to the HT. This payload can be categorized into the following threats:

1) Functional modification of the IC.
2) Specification modification of the IC.
3) Information Leakage
4) Denial of Service

Hardware Trojan implementation can perform any or all of these malicious actions [WANG2008]. We illustrate these threats in the following.

5.2.5.1 *Functional Modification*

A Hardware Trojan modifies IC functionality by adding, removing or bypassing existing logic. In the case of a RFID tag, this modification can be for example the password modification. In [Karri2010] the HT is designed to alter the CPU instructions by changing the contents of the program memory.

5.2.5.2 *Specification modification*

This class of Hardware Trojan can perform other malicious operations as modifying system clock or introducing timing errors. The work [Bhunia2010] presents an example of Hardware Trojan that introduces a fault by the insertion of resistor and capacitor to increase net delay.

5.2.5.3 *Information Leakage*

This payload presents a great threat for RFID security applications. The aim of this class of Hardware Trojan payload is to transmit sensitive information to adversary. The channel can be a wired connection as RS232, JATG, or a wireless connection as proposed in [Jin2010]. This wireless connection leaks the encryption key by manipulating the transmitted signal.

5.2.5.4 *Denial of Service*

The Hardware Trojan can be involved in low-level operations. This gives him the capability to downgrade or completely disable a system. An example of Hardware Trojan of this class appears in the work presented in [WOLF 2008]. The Hardware Trojan does not let the system going into the sleep state. Then the system consumes more power, thus limiting its service life.

A Hardware Trojan can also be used to modify an existing value in the memory by a random value, and this will cause an error in the services done by the circuit [Karri2010]. In addition, in [Bhunia2010], the authors provide an example of HT that generates an excessive activity which accelerates the aging process of an IC and thus shorten the device's life.

We have described the Hardware Trojan taxonomies for both trigger and payload. In the next section we will show methods and approaches to detect the HT.

5.2.6 *Hardware Trojan detection approaches*

The objective of Hardware Trojan detection approaches is to determine if an IC is infected or not. If the Hardware Trojan detection approaches do not detect any HT, we cannot conclude that the IC is trusted or the design team is trusted. Actually it is very hard to prove that an IC is Hardware Trojan free.

In the following, we will discuss some popular Hardware Trojan detection approaches and mechanisms. Each one will address a special type of Hardware Trojan. Since there are several types of HT, we need a detection methods classification such as the one proposed in [Mark2011] represented in Figure 5.8.

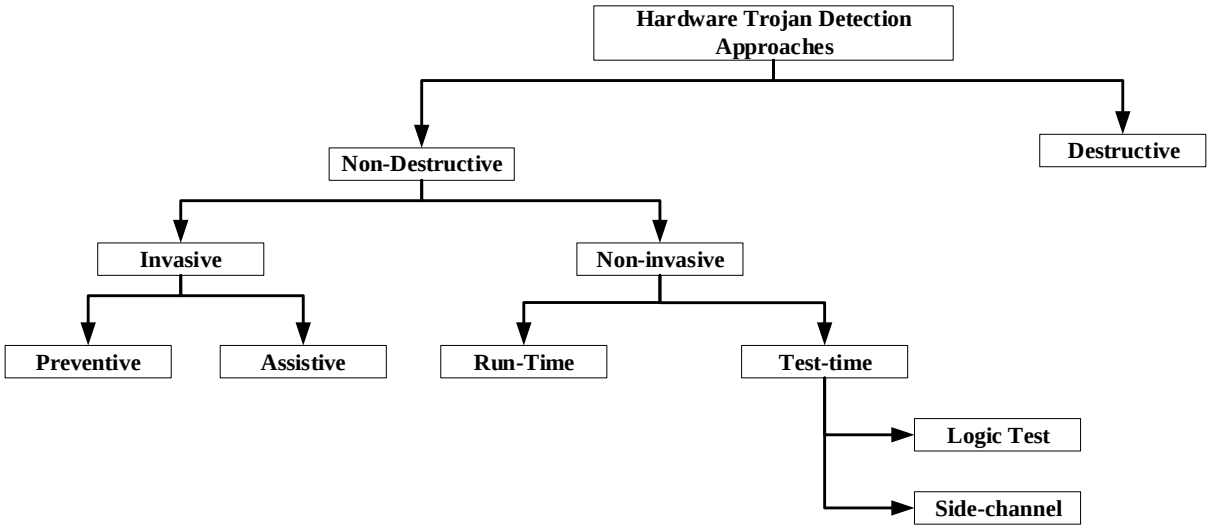


Figure 5.8 : Hardware Trojan Detection Techniques [Mark2011]

We will briefly present some proposed detection mechanisms, and we will focus on the methods that could be used in the particular context of RFID systems.

5.2.6.1 *Destructive detection approaches*

To ensure that there is a Hardware Trojan in a given IC, the IC can be completely disassembled and destroyed. Using this approach the HT will be detected by reverse-engineering. But this is a complex approach, which is also time consuming and expensive. This approach has other limitations. Hardware Trojan consists in a few logic gates added to the original circuit that generally contain a lot of gates. Thus it can be very difficult to find the modification, and this requires a deep reverse engineering study.

5.2.6.2 *Non-Destructive detection approaches*

This approach, which does not destroy the IC, is classified in two categories: invasive or non-invasive. Non-invasive techniques do not change the design of IC, while invasive techniques modify the design in order to embed functionality helping the detection of Hardware Trojan.

5.2.6.2.1 Invasive detection approaches

Invasive techniques are divided into two branches, prevention and assistance.

a) Prevention detection approach

Prevention approaches are used to prevent the insertion of Hardware Trojans prior to fabrication; this is done by controlling all the processes of IC fabrication. This approach can be

performed only if we ensure the use of trusted EDA tools, and if we fabricate the IC in a trusted foundry….etc. Today this approach is often impractical.

b) Assistive approach

Assistive approaches are based on the principle of addition of inputs/outputs to facilitate the detection of Hardware Trojans. In work [Chakraborty2008] the authors propose a design to detect Hardware Trojan in a system which contains several modules as an ALU, address decoder and memory. They add an input and an output to each module. These additional entries give the possibility to put the module in "transparent" mode. In this mode, the modules perform a self-test to detect rare events. The result is a signature, which is a combination of the provided input key and the results of its self-test as depicted in Figure 5.9

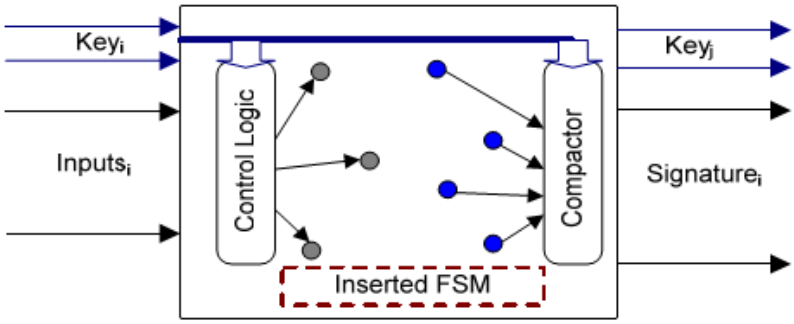


Figure 5.9: Assistive approach to detect Hardware Trojan [Chakraborty2008]

In the work [Salmani2009] several dummy flip-flops are inserted into logic. They aim at increase Hardware Trojan activity to facilitate detection using side-channel techniques.

5.2.6.2.2 Non Invasive

In this approach Hardware Trojan detection is done by comparing a golden and an infected copy of the IC. This approach can be done either at run-time or at test-time.

a) Run-time detection:

The idea of this approach is to monitor the IC behavior in real time. If the IC contains Hardware Trojan and the HT is activated, malicious behavior can be detected. It is preferable to add an interrupt mechanism coupled with the run-time monitoring system. If the Hardware Trojan is detected the interrupt mechanism stops the execution of all operations done by the IC. This will prevent the IC to perform malicious operations. The Run-time monitoring approaches need an additional embedded monitoring unit. This will take additional resources as power consumption or processing time and can downgrade the performance of the IC. The authors of

[Abramovici2009] added a reconfigurable unit in the IC. This unit performs the online monitoring of all operations done by the IC. They called this unit DEFENSE (DEsign-For ENabling-Security). This unit can detect in real time the change of behavior of a circuit.

Only the IC manufacturers can use these methods because these methods require design modifications. In the context of RFID, these methods cannot be used by the users which buy an already implemented ASIC (tag IC).

b) Test time

b.1) Side channel analysis

This approach is based on side channel information monitoring when the IC executes operations. It consists in monitoring a physical parameter as power consumption, temperature, EM radiation… etc. Any abnormal variations in the monitored parameters mean an additional operation in the IC and so a Hardware Trojan detection .Using such a method we can detect the Hardware Trojan but we cannot know its functionality. In [Banga2009] the authors propose a side-channel technique that is able to find differences in the power consumption between infected circuits and those that are not infected.

b.2) Logic testing

In the logic testing based Hardware Trojan detection, different test patterns will be applied to the IC for the purpose to activate the Hardware Trojan.

This approach encounters the problem of the full test coverage. For example, if a combinational IC has n inputs it requires 2 power n test vectors. Moreover sequential IC will require more test vectors. Thus, the amount of data in order to perform exhaustive test is not practical. To solve this problem, a random test can be used as presented in [Jha2008]. But this does not guarantee the Hardware Trojan detection success, since we cannot guarantee the Trojan has been triggered.

Several approaches are used to detect the presence of Hardware Trojan. Ones that do not require the use of golden models as the invasive approaches based on reverse engineering, or the ones based on the real time event monitoring. Other approaches require the use of golden models as the logic testing approach. Many approaches require activating the Hardware Trojan to detect it but few approaches do not even need the HT activation to detect it (for example, detection based on side channel approaches). Many detection approaches require first to modify

the design (to add control and monitor capability) and thus can often only be used by the designers.

5.3 Hardware Trojan in RFID System

In this second part of the chapter, we first present Hardware Trojans in the context of heterogeneous RFID systems, then present how the previously described hardware emulation platform can help to evaluate their risks and their countermeasures. In the following, we focus on the specific case where an infected IC has been designed incorporating an untrusted digital baseband IC. In other words, the Hardware Trojan has been inserted during the early IC design phases, and then all the chips are infected.

Figure 5.10 hereafter gives a simplified life cycle of an RFID tag. First, the tag is designed (most of the time by fabless companies), then the integrated circuit (IC) is fabricated and tested. Finally the IC is integrated with the antenna before to be personalized and deployed in a tracking system.

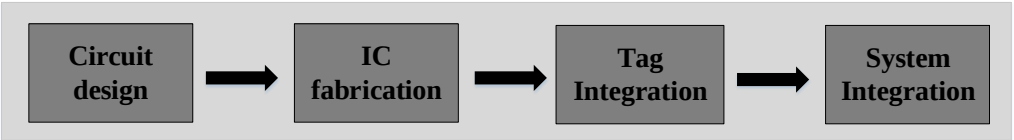


Figure 5.10 : RFID IC tag Lifecycle

The attacker can be either the chip provider, an IC designer, or a third party IP provider. It is to be noted that in this case, it is hardly possible to detect the malicious function since there is no golden reference circuit to compare with. So The RFID tags using this digital baseband are then infected and providing a backdoor to attackers in order to jeopardize any traceability systems using such tags (as represented in Fig. 5.11).

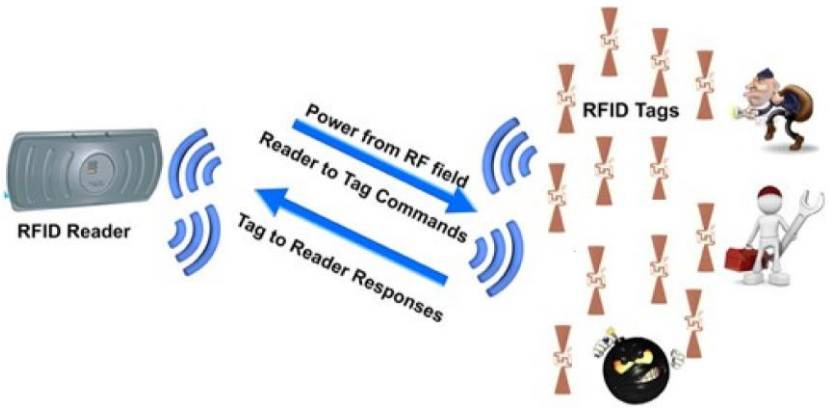


Figure 5.11: Malicious tags in an infected RFID System

In the following, we show how a Hardware Trojan can be embedded within a simple RFID tag. Using the emulator we discuss the specificities of RFID system which have to be managed by the attacker. Then we discuss how the HT can be exploited by attacker within the system.

5.3.1 *Objective of Hardware Trojan in RFID*

Hardware Trojan payloads within RFID systems are discussed below. An attacker willing to insert a malicious function within such an integrated circuit has then many options. Let's first consider the payload (i.e. the malicious function which will be activated in mission mode). According to RFID systems characteristics described earlier, we can consider different types of payloads as described on Figure. 5.11.

5.3.2 *On-field maintenance backdoor*

The HT in this case just aims at providing additional functions either for the chip or the tag provider. These additional functions consist in modifying the tag FSM by adding new commands. These new commands not included in the standard will give an advantage to the provider in order to perform on line monitoring of the tag (memory test, configuration checking etc.…). Such functions could mainly be used for system protection against tag tampering.

5.3.3 *Denial of Service Attack*

A major threat of Hardware Trojan in such IC would be a malicious function which is able to block all the communications between several tags and readers in the same area. As discussed earlier, UHF RFID tag are mainly used for traceability purpose, thus if one tag is able to block the communication even for a short time, many objects could pass through a traceability check. This can simply be done by blocking the tag in a "communication state". If this infected tag continuously backscatters the reader signal, this will make collisions with other tags attempting to communicate with the reader. Although the EPC C1 Gen2 protocol is designed to fix collisions, in such a case it is not efficient since the tag is blocked and does not fulfill anymore the protocol.

5.3.4 *Data eavesdropping/modification*

One other threat concerns the data stored within the IC. Even if UHF RFID tags are not designed for high end security applications, they may store confidential data (concerning the traceability of the tagged object…). Moreover the EPC C1 Gen2 standard is evolving in order to provide secure authentication to protect the user memory. The tag architecture may be

modified in order to remove the memory protection. So once the Hardware Trojan has been trigged, one can access to the protected memory without authentication. Also, a simple HT can leak a confidential data by replacing its RN16 number by either a password or a key.

More complex attacks have also to be considered, the tag can be used to corrupt the system middleware as suggested in [Mitrokotsa2010] a malicious circuit within the tag could dramatically ease the attack feasibility.

5.3.5 *RFID System specificities*

As discussed in the previous sections, RFID systems are strongly heterogeneous. They combine digital and analog hardware, software (within the reader to manage communication between tags and readers) network management (most of the time readers are connected together and managed by a middleware dependent of the application ([Floerkemeier2005], [Welbourne2007]) and electromagnetic waves involved in the tag communication and power. When evaluating Hardware Trojan threats against such systems it is important to consider all these system components. One should thus be able to evaluate:

- The hardware: how the HT can be embedded within the tag (considering tag performance, tag functionalities)
- The interaction between the tag and reader: does the tag operate properly when the HT is silent? How the HT can be activated by a reader?
- The RF channel: Is the HT robust to RF perturbations? What is the effect of the HT against other tags to reader communication?
- The software: How the software behaves against HT? Can we develop dedicated software to thwart such threats?

While register transfer level (RTL) simulation is a good candidate in order to evaluate a HT from a hardware point of view. It raises several issues when considering the whole system. First performances are too low to be able to simulate a complete inventory with several tag instances. Second, it is not possible to consider the possible errors within the RF channel and it is not adapted to consider the software parts. In [Fritz2010], the authors propose a SystemC based simulator. Such a simulator has the main advantage that it is can mixed hardware and software simulation and it is thus particularly adapted to simulate complex systems. The main drawback here is that the interactions within the RF channel are too simplified in the model to

accurately describe all potential effects which interest us in the particular case of Hardware Trojans.

For all these reasons, we consider an emulation based platform is the ideal tool in order to evaluate Hardware Trojan threats in EPC C1 Gen2 systems. The emulation allows validating the compliance of the emulated architecture against the standard by using a real reader. Also the emulator can be used within a complex system (as depicted in Figure 5.12) and all the perturbations within the RF channel can be taken into account. This is important for instance to be sure a perturbation will not accidentally activate the Trojan. Finally thanks to the emulation, all the system elements can be considered: the Hardware Trojan triggering mechanism, the effect on the system, etc.

Figure 5.12: RFID Emulation based platform for online monitoring of tag

Using the developed emulator, we can create and evaluate the feasibility of several Hardware Trojan scenarios with several types of trigger and payload. For the payload the possibility are numerous, but the consequences are always the same: leak of information, Denial of Services or functionality or specification modifications.

As shown in the Figure 5.13 below, we use two copies of our emulator to disturb the other tags. One of this emulator is used to inject the HT in the system. The other is used to monitoring the system behavior.

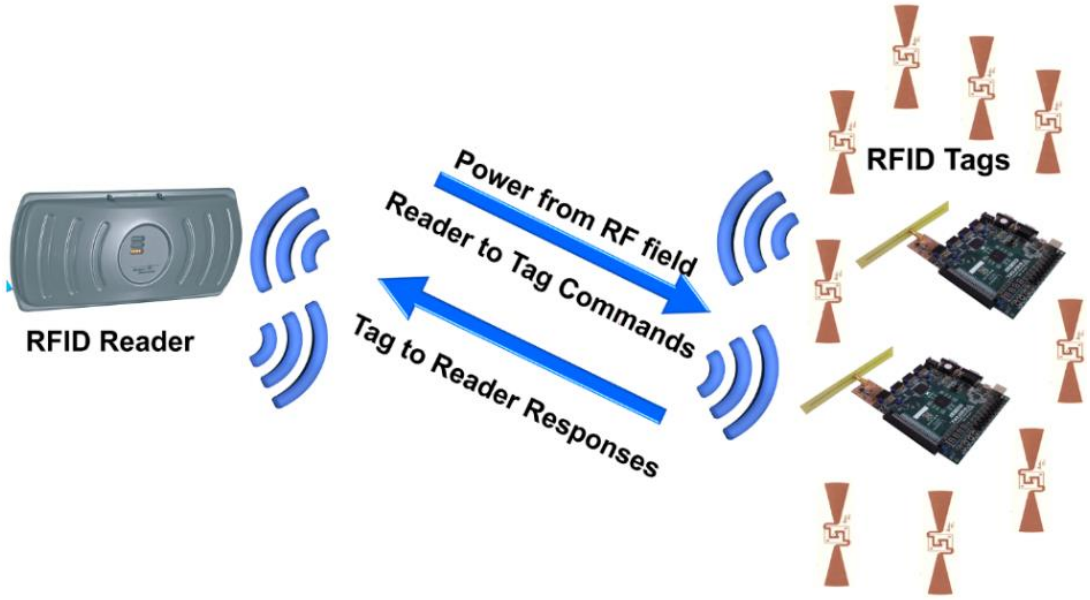


Figure 5.13: Hardware Trojan in System Emulation

5.3.6 *Trigger design and validation*

5.3.6.1 *Triggering issues*

In this section we focus on how a trigger mechanism can be designed in order to activate the Hardware Trojan within an EPC system. The trigger design of an RFID IC may fulfill two important characteristics:

- First, RFID ICs are standardized and have to be certified by a third party tester. The Hardware Trojan should thus be designed in such a way, the IC is still compliant with the standard and it is not too sensitive so that it cannot be trigged during certification testing.
- Secondly, RFID system communications are often disturbed due either to the presence of other tags, radiations or perturbations in the RF communication channel. So the modified design must be robust against all these potential perturbations.

An essential characteristic of Hardware Trojan is to switch on a very rare or temporary condition, in order not to be detected during functional tests [Wolff2008][Chakraborty2009]. In RFID system, system monitoring is a common practice to detect any system failure or error, and then monitoring could detect very rare command or strange sequence of commands corresponding to Hardware Trojans triggering. This makes trigger design more challenging since it has to pass both functional test (i.e. test done by the system integrator) and online monitoring. Hardware Trojans proposed in the following should then be resilient against possible detection during:

- The circuit verification stage and integrator functional test.
- The on line monitoring of RFID system

Usual functional tests made on RFID tag are depicted in Figure 5.14, such tests are based on EPC commands exchanged between the tag and a reader. The Query command initiates the communication and set parameters such as the modulation, the data rate and the rest of the parameters of the communication [EPC2008]. During functional testing, each command is tested with different parameters which could accidently activate the Hardware Trojans.

On line monitoring of the system is based on spying the communication between the reader and the tags as illustrated on Figure 5.14. Then the data exchanges are evaluated in order

to detect any abnormal behavior or abnormal performance degradation. For instance a usual RFID application is the tags inventory which consists in listing all the tags present in the field of the reader. The inventory is made accordingly to a specific protocol in order to fix collisions, and then once a tag has been inventoried, the reader can communicate specifically with one tag in order to exchange specific data. All these operations have to fulfill the standard. It is thus possible to monitor the system just by spying the command and the sequence of commands send by the reader to detect any abnormal behavior due to Hardware Trojan activation.

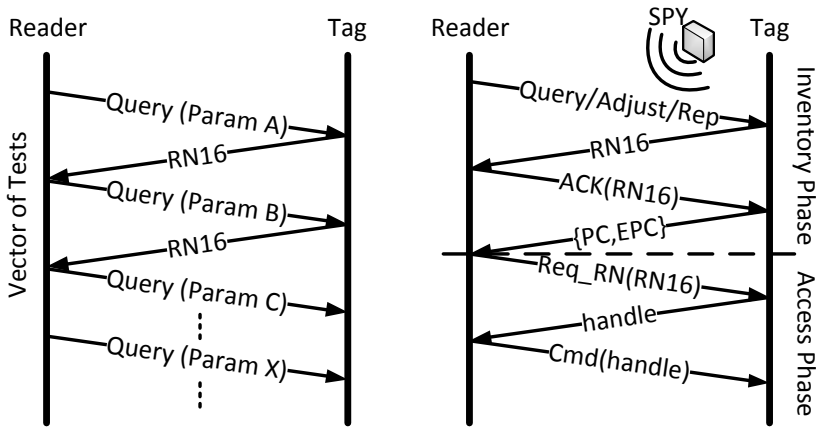


Figure 5.14: Example of tag Reader communication

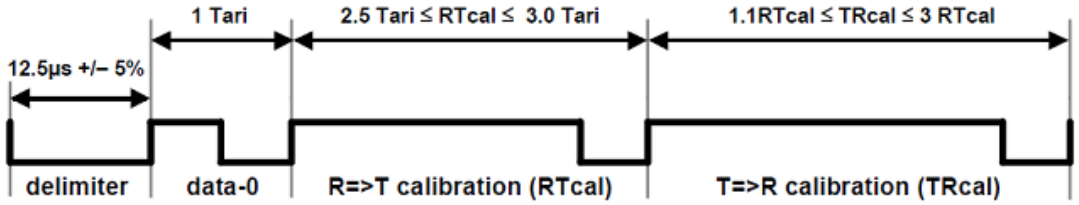
In the following, we propose three different types of trigger sensitive to:

1) Communication parameters.
2) Communication frame content
3) Sequence of commands

5.3.6.2 *Triggers sensitive to parametric changes (HT1)*

The EPC C1 Gen2 protocol specifies the frame parameters for the low level communication. For instance, the frame begins with a preamble and synchronizing elements like the Delimiter, Tari, RTCal or TRcal [EPC2008] (Figure 5.15-a). We can use specific values of these EPC parameters for activating Hardware Trojan. For example, as shown in Figure 5.16- b the HT will be activated if the tag receives a frame with a given RTCAL parameter value.

a)



b)

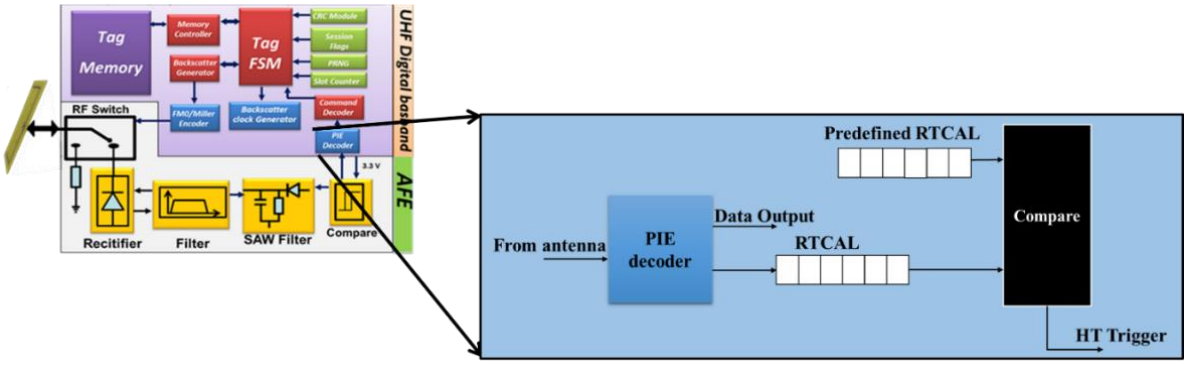


Figure 5.15 : a) Reader to tag preamble frame,

b) HT trigger design

5.3.6.3 *Triggers sensitive to a sequence of commands (HT2)*

This method consists of activating the Hardware Trojan when the tag receives a specific and valid but unlikely command in the current state of the Finite State Machine (FSM). For illustration the trigger HT2a has been designed so that the HT is activated once the tag receives a BlockErase command while the FSM is in the Reply state Figure 5.16.

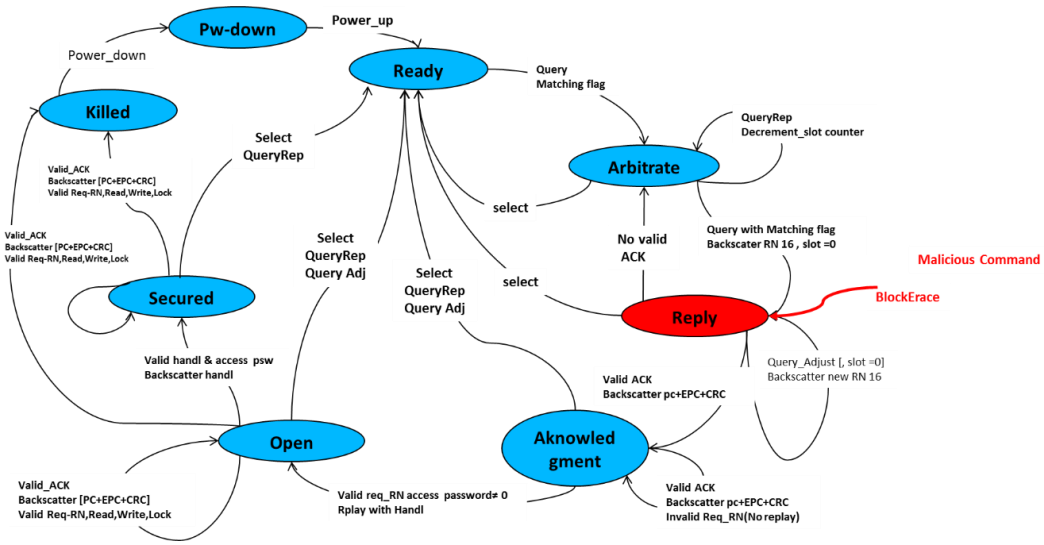
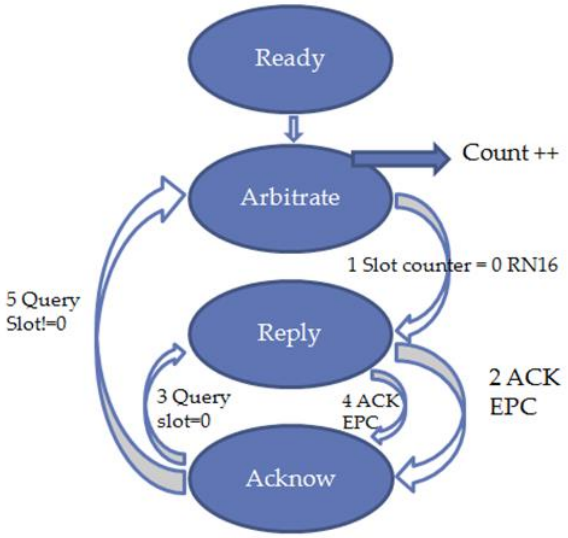
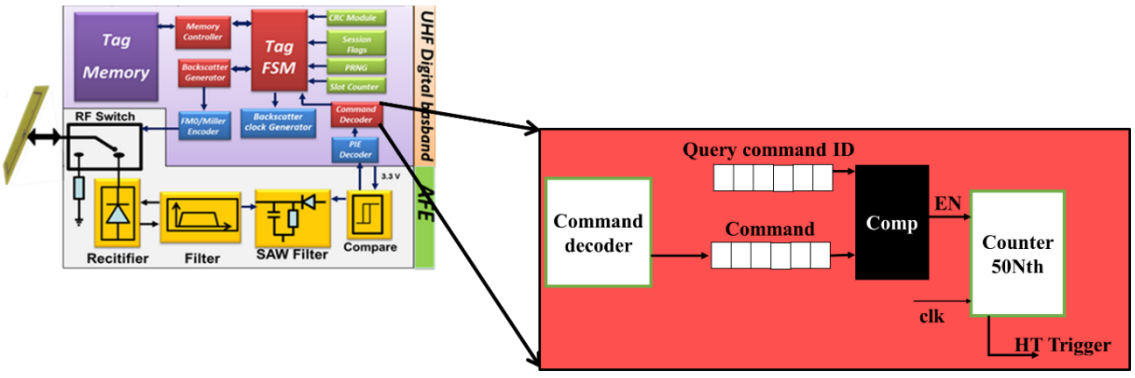


Figure 5.16: Hardware Trojan HT2a trigged by BlockErase command

Another similar approach lead to design a trigger activated when the tag receives a high number of times an unlikely command for a given FSM state. For example, similarly to the above trigger, if the tag receives 50 times a Query command while the FSM is in the Reply state as shown in Figure 5.17 a) and b), the trigger will be activated. This kind of trigger (HT2b) is more difficult to detect during functional test. During functional test, the probability that the command is sent such a number of times is very low. However, from a hardware point of view this trigger requires more logical element, increasing then the tag cost and consumption which is a major concern for such IC.



a)



b)

Figure 5.17: a) FSM modification b) Hardware Trojan trigger HT2b design

In order to increase the difficulty to functionally detect the Hardware Trojan, a solution consists in developing a trigger activated by a dedicated sequence of command. This sequence can be as complex as desired, taking into consideration that it requires to add a small FSM into

the circuit. Using the emulator, we have implemented a trigger (HT2b) which is activated after a given sequence of commands has been repeated X times by the reader.

5.3.6.4 *Modification of frame contents (HT3)*

In order to reduce the detection risk during functional test we have developed a trigger (HT3) based on the modification of the low level information present in the data frame. In other words, it intends to modify the frames sent by the reader to the tag. While on line monitoring checks the sequences of command exchanged in the system, bit level testing is more permissive since bit errors may be due to harsh environment.

Moreover during functional test, all set of commands are tested but it is hardly possible to test all command parameters. This gives thus room for adding a trigger based on bit level modification of the data frame. A generic EPC C1 Gen2 frame is made of the command-ID, the parameters and an integrity check (i.e. based on cyclic redundancy code [EPC2008]),as shown Figure 5.18.

Figure 5.18: EPC C1 Gen2 Frame [EPC2008]

Modifying the frame content induces then modifying bits in these fields of the frame. We can thus either modify the Command-ID (first bits in the frame) or the parameters (in the middle of the frame) without modifying the CRC (last bits in the frame) or doing right the opposite, modifying the CRC without modifying the rest of the frame. Therefore the CRC will not be correct and the frame will be deleted when the tag checks it.

According this principle two HT have been designed. As described earlier, a CRC is computed with the Command-ID and the parameters. A way to silently trig the Trojan is to corrupt the parameters so that:

1. When the tag calculates the CRC, if it does not corresponds to the one sent by the reader, then the command is discarded.
2. The CRC result of the tag computation corresponds to a golden value saved in the tag which trigs the Trojan.

Such a trigger (HT3a) requires little change in the tag design and adds very few components. Moreover, it is very hard to detect by on line monitoring since it can be associated to bit failure in the communication.

Another mean is to modify the CRC computation (HT3b). When a frame arrives to the tag and the result of the CRC checking is wrong, it is computed again taking into account that the command-id + parameters could have been modified (Figure. 5.19). Therefore, in the new checking of the CRC a dedicated data is added to the command and the parameters. If the new result of CRC is right, the HT is switched on. As it is expected, this implementation requires more logic elements but it is safer (it has a very low probability to be accidentally trigged) than the previous approaches.

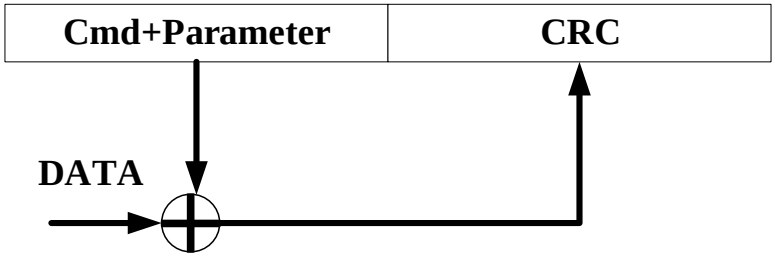


Figure 5.19: HT3b design, CRC calculation with a malicious data

5.3.7 *Fault Injection based Validation of Hardware Trojans*

As discussed earlier, one difficult thing about inserting Hardware Trojan within RFID tag is the possibility to master the activation time. Since they are used in harsh environment, they may be sensitive to perturbations inducing internal single or multiple event upsets or the communication may be perturbed by collision so that the tag receives corrupted data. In both cases internal errors or corrupted data may lead to an unwilling Trojan activation. Using injection fault capabilities of RFIM provides a powerful tool to validate in situ the HT robustness. The emulator can be placed in front of a commercial reader with other commercial tags. Then while the reader does the inventory, the injection block inserts randomly errors within the digital baseband. These errors can be sent in the command decoder, the data frame, or any registers of the design. Doing so we are able to emulate any communications errors or on chip malfunction which could happen. Then thanks to online monitoring capabilities, any activation of the Hardware Trojan under validation can be trapped.

5.3.8 *Triggers Evaluations*

We have implemented all the triggers within RFIM. First using characterization tools, we have validated that the tag with the Trojan still fulfils the EPC standard. Then the different RFIM capabilities discussed above have been used in order to evaluate the Trigger (1) robustness and (2) discretion. Table 5.2 hereafter gives the pros and cons for each proposed triggering mechanisms. They are evaluated according three main characteristics, the design overhead induced by the malicious function, the robustness of the mechanism measured by its sensitivity to on-chip errors and finally the discretion which indicates the capability of the trigger to be activated without inducing suspicion. We have discarded HT1, since obviously this kind of trigger is too sensitive to RF perturbations and have a high probability to be accidently activated. While discretion must be compromised with robustness and design overhead, HT3b appears as the one reaching the best compromised.

Table 5.2: Triggers Evaluation

HT	Design overhead	Robustness	Discretion
HT1	NA	NA	NA
HT2a	+	+	--
HT2b	--	+	-
HT3a	-	--	+
HT3b	-	-	++

5.3.9 *Payload design and validation:*

Once the Trojan is activated the payload is the action or damages applied to the RFID tag. There are three major types of damage.

1) Denial of service
2) Leak sensitive information
3) Modify the functionality or Specification

This section focuses on the damage of a payload. We take advantage of our emulation platform RFIM to show example of damage caused by Hardware Trojan in RFID system.

5.3.9.1 *Denial of services HP1*

The payload modifies sensitive register (parameter) by a random number from PRNG generator as described in Figure 5.20. This parameter can be: TRCAL, DR, Query session. If an error is introduced in this parameter, this causes a wrong computation and the tag does not

response to the reader. Table 5.3 summarized the effect or damage that can be caused by this Hardware Trojan.

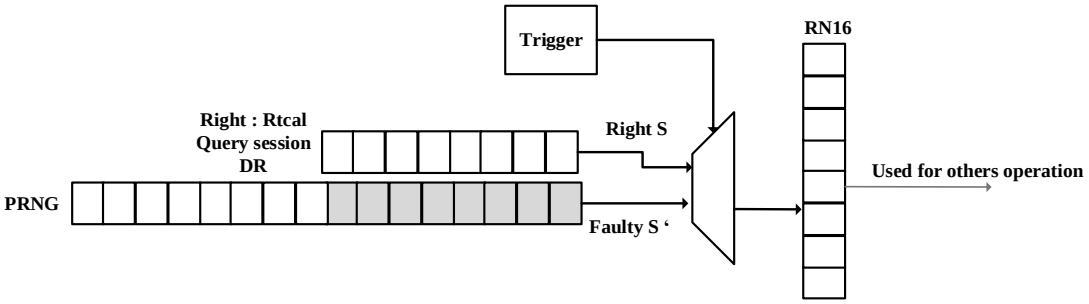


Figure 5.20: Payload HP1 Modify Specification (parameters)

Table 5.3: HP1 payload effects of RFID tag

Parameter	**Effects**
Query session	Decrease the Read Rate of the tag
Trext	Response encoding error
DR	Response encoding error
RTCAL	Decrease or increase response time (time out)

5.3.9.2 *Leak sensitive information HP2*

This payload can be the most dangerous. As we have seen in the previous chapters, the tag memory stores highly sensitive information such as tag ID, cryptography key, etc. Using this payload, the tag sends these information using the normal responses to the Query command. But instead of sending the RN16 it sends secret sub-parts of information. As seen in Figure 5.21, each secret information is sent in packet of 2 bytes, since the size of RN16 is 2 bytes. The payload HP can leak the following information:

- kill password
- access password
- Crypto key

The role of the Counter is the organization of the leak of information. Each time the counter trigged, Hardware Trojan passes information. When counter equals to zero 0 the payload leaks the first half of kill password, if counter equals to 5 the payload leaks the tag identifier, and so one.

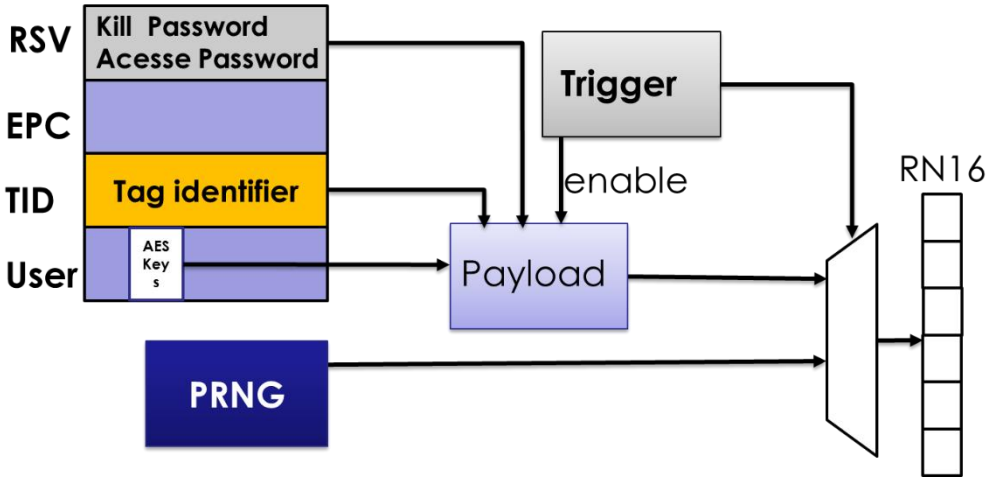


Figure 5.21: Payload HP2, leak of information

5.3.9.3 *Modify functionality HP3*

As we have seen in the last chapters the tag contains 4 banks of memory [EPC2008]. This memory is protected by a password of 32 bits. The RFID reader cannot read or write in this tag memory, if it does not provide the right password and after the tag checks this password.

The payload HP3 disables the mechanism of password verification as described in the EPC standard and thus the tag accepts any password to access to its memory (Figure 5.22).

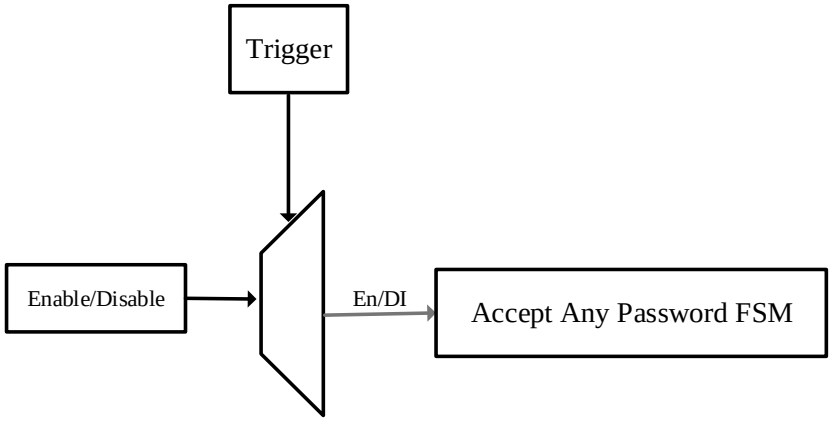


Figure 5.22: Payload HP3 Disable the mechanism of password verification

5.3.9.4 *Modify specification HP4*

As we have seen in the last chapters that EPC C1 Gen2 standard defines communication parameters used by tag during its response to the reader (backscattering). The value of backscatter link frequency (BLF) period is defined by the length of the TRcal parameter and a Divide Ratio (DR) parameter (with value 8 or 64/3).

The malicious operation of the payload HP4 is the modification of the BLF. This involves a modification of the bit rate and desynchronizes the "reader tag link" as depicted in Figure 5.23. Due to this HT the reader will ignore the tag response, and the attacker will have the opportunity to intercept the tag response and to take over to communicate with infected tag using another reader. Figure 5.23shows that the payload can change the DR and TRCAL parameter to different fixe values.

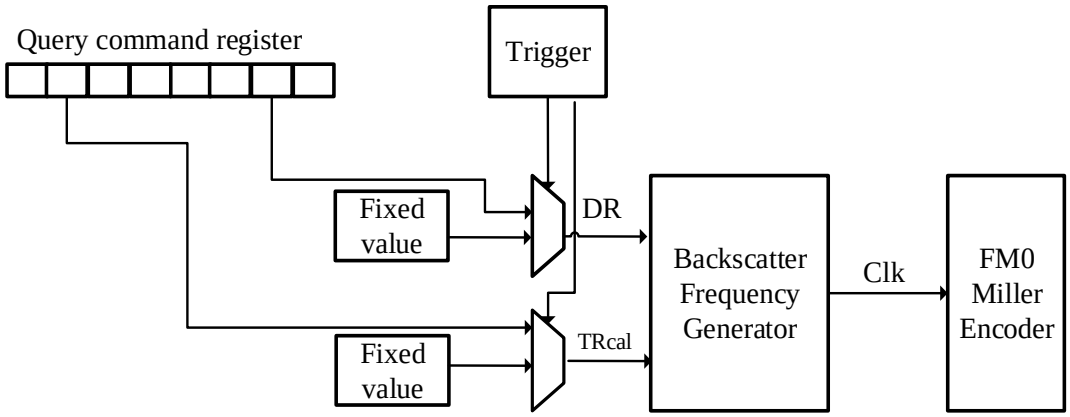


Figure 5.23: Payload HP4, specification modification

5.4 On line monitoring for Hardware Trojan Detection:

So far, the proposed emulator has been used as a validation tool. Thanks to its online monitoring, it could be used as a system spy in order to detect any suspicious operations. The emulator can be placed in the critical parts of the system and monitors the data received by the tag (these data can come from either a corrupted reader belonging to the system or an external reader introduced in the system by an attacker). In that case, the dedicated part to the injection fault is not used. The emulator registers all the data being in RF field. Then the on board microprocessor analyzes the coherence of the data which have been sent and can then raise a flag to the system in case something suspicious has been detected. As an illustration, the microprocessor runs a software checking that the sequence of command seen by the tag corresponds to a meaningful one in order to detect HT like HT3 and HT4. Nevertheless, even if this method can easily detect HT, the microprocessor may be too slow and then flag rising may arrive too late to the system supervisor. In order to solve this problem, the emulator has been enhanced in order to incorporate on line checker within the digital part of the tag itself as proposed in [Omar2013]. OVL properties are embedded within the emulator in order to check the coherence of the data processed by the tag. Verification properties can be easily edited in

order to detect Hardware Trojan like HT3 and HT4. The main advantage here is that it does no more require a post treatment from the embedded processor and then accelerates the detection.

There is a strong similarity between a fault and the effect of Hardware Trojan. Table 5.4 compares the properties of Hardware Trojans with those of faults.

Table 5.4: Comparison between fault and Hardware Trojan attacks

	Fault	**Hardware Trojan**
Activation	Usually at known functional state	Arbitrary combination /internal circuit sequence (digital or analog)
Insertion agent	Accidental	Intentional
Manifestation	Functional /parametric failure	Functional/parametric failure and information leakage

Unlike faults in IC that occur due to harsh environment or due to an error during manufacturing, Hardware Trojans are intentionally inserted by adversary to serve a specific malicious purpose. A fault is activated for a known functional state (or for a known input value) of an IC. For example, a stuck-at-0 fault at the output of an inverter is activated by controlling its input to a 1. At contrary a HT is designed to be activated at an arbitrary condition. However, as the effects of fault and HT are similar, we can use the same countermeasures used for fault detection to detect the HT. Thus, using Error Code Correction allows detecting numerous modifications (due to HT or fault) in any registers of an RFID tag. In addition, using OVL we can also monitor the internal behavior of tags and detect misbehaviors of RFID tags when the tags are read by a reader. Using this method, we can detect the tag infected by a Hardware Trojan as the misbehaviors due to faults.

5.5 Conclusions

In this chapter, we have first introduced the potential effects related to Hardware Trojans in EPC C1 Gen2 RFID systems. We have shown from an attacker point of view how could be designed a trigger to activate the Hardware Trojan. It is important to take into account the heterogeneity of RFID systems and their use cases in order to design a trigger which is robust and difficult to detect. Also, we have proposed an EPC C1 Gen 2 tag emulator which can be used to evaluate security hazards within RFID system and notably Hardware Trojan ones. The emulator has been used to validate the proposed trigger in front of industrial RFID readers. Finally, this emulator thanks to its online monitoring capabilities can be used as a system

countermeasure in order to spy RFID systems in critical stages and then to detect any suspicious activities.

6 Conclusion and perspective

6.1 Conclusions

This thesis addresses the design and the emulation of robust UHF RFID tags. Indeed, RFID technology is more and more used for critical applications within harsh environments or for secure applications in military, medical or security, in which the robustness of UHF RFID tags is crucial.

The chapter 2 of this thesis was dedicated to an overview of RFID technology. The various existing RFID technologies have been introduced. The structure of RFID systems has been detailed by explaining different technical details on reader architecture, tag architecture, and on the protocol used. In the second part of this chapter, we have shown the use of RFID technology in different critical applications.

In chapter 3, we have shown different methods to perform early analysis in the RFID system design flow. We have presented several RFID simulators with their advantages and limitations. Then we have discussed about the existing RFID emulation platforms. Finally a table comparing the advantages of each platform has been given. In the second part of this chapter, we have discussed about the robustness analysis of RFID system. This analysis is performed thanks to the analysis of the faults effect on this system. We have discussed about the different types of fault injection.

In chapter 4; we have presented the developed RFIM platform. We have shown how this platform helps to study the robustness of UHF EPC C1 Gen2 RFID tags in the presence of faults in a real environment. This study has shown that the most sensitive parts (registers) of a tag and its most sensitive parameters (parameter involve in the EPC C1 Gen2 protocol). This sensitivity analysis has helped to develop a low cost and more robust and secured tag architecture by the proposal of different solution and countermeasure tailored according to the fault injection results. We have shown how RFIM platform permits to validate the tag architecture in real environments.

In chapter 5, we have first introduced the potential hazards related to Hardware Trojans in EPC C1 Gen 2 RFID systems. We have shown from an attacker point of view how could be designed a trigger to activate the Hardware Trojan. It is important to take into account the heterogeneity specificities of RFID system and their use cases in order to design a trigger which is robust and difficult to detect. Also, we have proposed an EPC C1 GEN 2 Tag emulator which can be used to evaluate security hazards within RFID system and notably Hardware Trojan ones. The emulator has been used to validate the proposed trigger in front of industrial RFID

readers. Finally, this emulator thanks to its online monitoring capabilities can be used as a system countermeasure in order to spy RFID systems in critical stages and then to detect any suspicious activities. At this time, more Trojans are being developed including the payloads, in order to perform system level attacks. These ones will then be evaluated in a complete RFID system (from tags to middleware) in order to (1) evaluate the capabilities of such attacks against the system and (2) develop some countermeasures at all level of the systems.

6.2 Perspectives

Today RFID is believed to be common technology in our life. Unfortunately, the technology has been integrated without considering all usage and associate security risks. RFID as Internet of things technology suffers security issues. RFIM is a good candidate to first evaluate and demonstrate security threats of RFID and then define appropriate solutions at all system levels to secure this technology.

Perspectives of this work are thus to use this platform to increase the security properties of RFID systems.

Main security solutions rely on cryptographic primitives. Nevertheless most usual ones do not fit the low resources constraints of RFID system (silicon area, power budget). Moreover, these new primitives must be compliant with the EPC Gen2 standard. RFIM platform can help to evaluate the suitability of theoretical primitives to real RFID systems.

For instance, this platform can be used to validate the design of new lightweight security mechanism such as authentication protocol to ensure the privacy and avoid threats like illegitimate reading of data, these mechanisms must supports forward and backward security, because the only authentication mechanism used by EPC C1 Gen2 is a pseudorandom number generator (PRNG) that is shared with RFID reader during inventory round. Lightweight encryption evaluation for RFID can also be performed with RFIM, whil most of the proposed algorithm are only theoretically evaluated. Also PUF based primitives have been considered as very promising for RFID technology. RFIM platform could be enhanced to embed such technology and then develop an end to end demonstration of a secure protocol based on this technology. Finally, RFID offers many opportunities for sensing in complex systems. RFIM can easily integrates all kinds of sensors to perform fast prototyping of RFID based sensing systems.

Reference

Reference

[EPC2008] EPCglobal, EPC radio frequency identity protocols classe-1 generation-2UHF RFID, protocol for communications at 860 MHz 960 MHz, version1.2.0, 2008.

[Laprie2004] J.-C. Laprie, "Sûreté de fonctionnement informatique: concepts, défis, directions", ACI Sécurité et Informatique, Toulouse, Novembre 2004.

[Johnson1989] Johnson, Barry W. "Design and Analysis of Fault Tolerant Digital Systems." Design and Analysis of Fault Tolerant Digital Systems.1989.

[Avizienis2004] A. Avizienis, J.C. Laprie. "Basic Concepts and Taxonomy of Dependable and Secure Computing." IEEE Transactions on Dependable and Secure Computing Archive, 2004: vol. 1.

[Arlat1992] Arlat, J. "Fault Injection for the Experimental Validation of Fault Tolerant Systems." Proceedings of Workshop Fault-Tolerant Systems. Kyoto, Japan: IEICE, 1992. 3340.

[Barbosa2012] Raul Barbosa, and Mario Zenha-Rela.”Resilience Assessment and Evaluation of Computing Systems” 2012, pp 263-281, Springer 2012

[Madeira2000] Madeira, H, D Costa, and M Vieira. "On the emulation of software faults by software fault injection." Proceedings International Conference on Dependable Systems and Networks. New York, NY: IEEE, 2000. 417-426.

[Maia2005] Maia, R, L Henriques, R Barbosa, D Costa, and H Madeira. "Exception fault injection and robustness testing framework: a case-study of testing." 2005.

[George2010(b)] N. George, C. Elks, B. Johnson, J. Lach. "Spatial Multi-Bit Soft-Error Tolerance in Logic." Dependable Systems and Networks Symposium. Chicago, IL, 2010.

[Andres 2008] D. D. Andres, J.C. Ruiz, D. Gil, P. Gil. "Fault Emulation for Dependability Evaluation of VLSI Systems." IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2008.

[Andres2006] D.D. Andres, J.C. Ruiz, D. Gil, P. Gil. "Run-Time Reconfiguration for Emulating Transient Faults in VLSI Systems." International Conference on Dependable Systems and Networks.2006. 291-300.

[Stott2000] D.T. Stott, B. Floering, D. Burke, Z. Kalbarczyk, R.K. Iyer. "Nftape: A Framework for Assessing Dependability in Distributed Systems with Lightweight Fault Injectors." IEEE, 2000. IEEE International

[Vargas2005] F. Vargas, D.L. Cavalcante, E. Gatti, D. Prestes, D. Lupi. "On the Proposition of an EMI-Based Fault Injection Approach." 11th IEEE International On-Line Testing Symposium (IOLTS 2005).Stevenson, Washington: IEEE, 2005. 207-208.

[KARL1991] J. Karlsson, U. Gunneflo, P. Lidén, J. Torin "Two Fault Injection Techniques to Test of Fault Handling Mechanisms", Proc. of IEEE Int. Test Conference (ITC'91), pp. 140-149, 1991.

[KARL1995] J. Karlsson, P. Folkesson, J. Arlat, Y. Crouzet, G. Leber, "Comparison And Integration Of Three Diverse Physical Fault Injection Techniques", Predictably Dependable Computing Systems, pp. 309-327, 1995.

[Selmane2008] N. Selmane, S. Guilley, and J.-L. Danger, "Practical setup time violation attacks on AES," in Proceedings of the 7th European Dependable Computing Conference. IEEE, 2008, pp. 91–96.

[Fukunaga2009] T. Fukunaga and J. Takahashi, "Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers," in Proceedings of FDTC 2009. IEEE, 2009, pp. 84–92.

[Agoyan2010]M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria, "When clocks fail: On critical paths and clock faults," in Smart Card Research and Advanced Application –CARDIS 2010, ser. LNCS, D. Gollmann,J-L Lanet, and J. IguchiCartigny, Eds., vol. 6035. Springer-Verlag, 2010, pp. 182–193.

[Choi1992] G.S. Choi, R.K. Iyer. "Focus: An Experimental Environment for Fault Sensitivity Analysis." IEEE Transactions on Computers, 1992: 1515-1526.

[Jenn1994] 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.Pasadena, CA, 1994. 66-75.

[DeLong1996] DeLong, T A, B W Johnson, and J A Profeta. "A Fault Injection Technique for VHDL Behavioral-Level Models." 1996. 24-33.

[Kwang-Ting1999] C. Kwang-Ting, H. Shi-Yu, d. Wei-Jin. "Fault Emulation: A New Methodology for Fault Grading." IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 1999: 1487-1495

[Lala2012]P. Lala, "Transient and Permanent Fault Injection in VHDL Description of Digital Circuits," Circuits and Systems, Vol. 3 No. 2, 2012, pp. 192-199.

[Silicon2005].http://www.silicon.fr/rfid-larmee-americaine-sy-met-13594.html

[Capgemini2003]http://shipping.neopost.com/sites/neopostid.com/files/PressRelease/File/cap_nid_projet_silria_ministere_defense_gb.pdf

[Andy2005] Andy Montador,"Verification Methodology for Standards-based IP &SOC", IP Based SoC Design Conference, 2005, France.

[Lopez2008] P. Peris-Lopez, J. C. Hernandez-Castro, J.M. Estevez-Tapiador, and A. Ribagorda. LAMED : A PRNG for EPC Class-1 Generation-2 RFID specification. Journal of Computer Standards & Interfaces, 2008.

[Elks 2005] Elks, C. "A Theory of Run-Time Verification." Charlottesville, VA: University of Virginia, Ph.D. Thesis, 2005.

[Honorio2011] Honorio Martín, Enrique San Millán, Luis Entrena, Julio César Hernández Castro and Pedro Peris Lpez. AKARI-X: a pseudorandom number generator for secure lightweight systems. IEEE 17th International On-Line Testing Symposium, 2011.

[Kalikinkar2011] Kalikinkar Mandal, Xinxin Fan and Guang Gong, A Lightweight Pseudorandom Number Generator for EPC Class 1 Gen2 RFID tags. RIM Seminar, 2011.

[Choi1992] G. S. Choi and R. K. Iyer, "FOCUS: an experimental environment for fault sensitivity analysis," in *IEEE Transactions on Computers*, vol. 41, no. 12, pp. 1515-1526, Dec 1992.

[Wyld2005] Wyld, D. C. RFID (2005): The right frequency for the government. A research monograph from the IBM Center for the Business of Government.

[Want2006]Want, R. (2006) An introduction to RFID technology, IEEE Pervasive Computing, 5(9): 25– 33.

[Impinj2007] Impinj. Inc, (2007) RFID communication and interference, White Paper, Grand Prix Application Series.

[Currie2008] Currie, I. A. and Marina, M. K. (2008) Experimental evaluation of read performance for RFID-based mobile sensor data gathering applications, inProceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia, Umea, Sweden, pp. 92–95

[Dobkin2007]Dobkin, D. M. (2007) The RF in RFID: Passive UHF RFID in Practice. Oxford: Elsevier-Newnes.

[Land2012] Landaluce, H, Perallos, A. ; Angulo, I. "A simulation tool for RFID EPC Gen2 protocol". Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on

[Palazzi2008]C. E. Palazzi, A. Ceriali, M. Dal Monte, "RFID Emulation in Rifidi Environment", in Proc. of the International Symposium on Ubiquitous Computing (UCS'09), Beijing, China, Aug 2009.

[Angerer2009]Angerer, C.; Langwieser, R.; " Flexible evaluation of RFID system parameters using rapid prototyping", RFID, 2009 IEEE International Conference on Digital Object Identifier: 10.1109/RFID.2009.4911188 Publication Year: 2009 , Page(s): 42 – 47

[Floerk2009]Floerkemeier, C.; Sarma, S.; "RFIDSim—A Physical and Logical Layer "Simulation Engine for Passive RFID " Automation Science and Engineering, IEEE Transactions on Volume: 6 , Issue: 1 Digital Object Identifier: 10.1109/TASE.2008.2007929 Publication Year: 2009 , Page(s): 33 – 43

[Fritz2010-b]G. Fritz, V. Beroulle, O. Aktouf, M. D. Nguyen, D. Hély, "RFID System On-line Testing Based on the Evaluation of the tags Read-Error-Rate", JOURNAL OF ELECTRONIC TESTING, (DOI: 10.1007/s10836-010-5191-6), pp 1-10, december 2010.

[Azambuja2008]Azambuja, M.; Marcon, C.; Hessel, F., "A Communication Protocol and Physical Characteristics Simulator for an RFID Sensor Environment," Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08. International , vol., no., pp.1093,1098, 6-8 Aug. 2008

[Mirowski2009] Mirowski, L.; Hartnett, J.; Williams, R., "Tyrell: A RFID simulation platform," Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on , vol., no., pp.325,330, 7-10 Dec. 2009.

[Khouri2004]R. Khouri, V. Beroulle, T.-P. Vuong, and S. Tedjini. Wireless system validation using VHDL-AMS behavioral antenna models: radio-frequency identification case study. 7th European Conference on Wireless Technology, 2004.

[Yifeng2004]Yifeng Han, Qiang Li, Hao Min "System Modeling and Simulation of RFID"Auto-*ID Labs at* Fudan University, Shanghai, P. R. China, 2004

[Fritz2012] Gilles Fritz « Simulation de fautes pour l'évaluation du test en ligne de systèmes RFID » thèse, université de Grenoble, 2012

[Hutter2008]Michael Hutter, Jörn-Marc Schmidt, and Thomas Plos. 2008. RFID and Its Vulnerability to Faults. In proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems (CHES '08).

[Angerer2008]C. Angerer, M. Holzer, B. Knerr, and M. Rupp, A fexible dual frequency testbed for RFID," in Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities, 2008.

[Roy2006]N. Roy, A. Trivedi, and J. Wong, "Designing an FPGA-based RFID reader," XCell Journal, vol. 2, 2006.

[Buettner2012]M. Buettner, "A flexible software radio transceiver for UHF RFID experimentation,"Availableftp://ftp.cs.washington.edu/tr/2009/10/U-CSE-09-10-02.PDF[Mar. 15, 2012].

[Sample2008]A.P. Sample, D.J. Yeager, P.S. Powledge, A.V. Mamishev, and J.R. Smith, "Design of an RFID-based battery-free programmable sensing platform," IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 11, pp. 2608-2615, 2008.

[Buettner2004]M. Buettner, R. Prasad, A. Sample, D. Yeager, B. Greenstein, J.R. Smith, and D. Wetherall, \RFID Sensor Networks with the Intel WISP," in Proceedings of the 6th ACM conference on Embedded network sensor systems, 2008, pp. 393-394.

[Li2012]T. Li, A. Borisenko, and M. Bolic, \Open Platform Semi-passive RFID tag",11th International Conference, ADHOC-NOW 2012, Belgrade, Serbia, July 9-11, 2012. Proceedings.

[Chen2011]L. Chen, S. Zhang, Z. Wang, and L. Li, "A New Simulation Platform for UHF RFID tag Development," in 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM). Sept. 2011, pp. 1-3.

[Feldhofer2010]M. Feldhofer, M. Aigner, T. Baier, M. Hutter, T. Plos, and E. Wenger,"Semipassive RFID development platform for implementing and attacking security tags," in 2010 International Conference for Internet Technology and Secured Transactions (ICITST), 2010, pp. 1-6.

[Donno2011] D. De Donno, F. Ricciato, L. Catarinucci, A. Coluccia, and L. Tarricone,"Challenge:towards distributed RFID sensing with software-defned radio," in Proceedings of the sixteenth annual international conference on Mobile computing and networking,2010, pp. 97-104.

[Catarinucci2011]L. Catarinucci, D. De Donno, M. Guadalupi, F. Ricciato, and L. Tarricone,"Performance analysis of passive UHF RFID tags with GNU-radio," in 2011 IEEE International Symposium on Antennas and Propagation (APSURSI), 2011, pp. 541-544.

[Athalye2011] Athalye, A.; Savic, V.; Bolic, Miodrag; Djuric, P.M., "A Radio Frequency Identification System for accurate indoor localization," Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on , vol., no., pp.1777,1780, 22-27 May 2011.

[Jun2010] Jun-Seok Park; Jin-Woo Jung; Si-Young Ahn; Hyoung-Hwan Roh; Ha-Ryoung Oh; Yeung-Rak Seong; Yoon-Deock Lee; Kyoung Choi, "Extending the Interrogation Range of a Passive UHF RFID System With an External Continuous Wave Transmitter," Instrumentation and Measurement, IEEE Transactions on , vol.59, no.8, pp.2191,2197, Aug. 2010.

[Dobkin2007]D. Dobkin, The RF in RFID: Passive UHF RFID in Practice, Newnes, 2007.

[Naray2010]G. Narayanaswamy, Blocking Reader: Design and implementation of a low-cost passive UHF RFID Blocking Reader, Ph.D. thesis, University of Texas, 2010.

[Ahson2008]Ahson, S. A. and Ilyas, M. Eds., (2008)RFID Handbook: Applications, Technology, Security and Privacy. Boca Raton, FL: CRC Press.

[Fritz2010-a]Gilles Fritz, Vincent Beroulle, Oum-ElKheir Aktouf , Minh Duc Nguyen,David Hély. RFID System On-line Testing Based on the Evaluation of the tags Read-Error-Rate, Mixed-Signals, Sensors and Systems Test Workshop (IMS3TW), 2010 IEEE 16th International

[iaik2014]IAIK TU Graz, RFID tag emulators for HF and UHF frequency range," http://www.iaik.tugraz.at", Last access date: 2014/03/01.

[Kenarangui2010]N. Kenarangui, Real Time Location Tool for Precision Tracking of Passive UHF RFID tags in Two Dimensions, Ph.D. thesis, University of Texas, 2010}.

[Abdelmalek2013]Omar Abdelmalek, David Hely, and Vincent Beroulle, Ibrahim Mezzah "An UHF RFID Emulation Platform with Fault Injection and real time Monitoring capabilities" 8TH IEEE INTERNATIONAL DESIGN & TEST SYMPOSIUM (IDT 2013).

[ABDELMALEK2014-a] O. Abdelmalek, D. Hély, V. Beroulle "Emulation of Faults Injection on UHF Transponders", in 17th IEEE Symposium on Design and Diagnosis of Electronic Circuit and System (DDECS 2014), Warsaw, Poland, 23-25 April 2014

[ABDELMALEK2014-b] O. Abdelmalek, D. Hély, V. Beroulle "Fault Tolerance Evaluation of RFID Tags", in IEEE Latin America Test Workshop (LATW 2014), Fortaleza, Brésil, 13-16 March 2014 doi:10.1109/LATW.2014.6841902

[Hidalgo2013] E. Hidalgo, O. Abdelmalek, D. Hély, V. Beroulle, "Triggering Hardware Trojans in EPC C1G2 RFID Tags" Workshop on Trustworthy Manufacturing and Utilization of Secure Devices, TRUDEVICE 2013, Avignon May 31st 2013

[Antoni2001] L. Antoni, R. Leveugle, B. Fehér, "Using Run-Time Reconfiguration for Fault Injection Applications", IEEE Instrumentation and Measurement Technology Conference (IMTC'01), vol 3, pp. 1773-1777, Mai 2001.

[Tombs2004] J.N. Tombs, F. Muňoz, V. Baena, A. Torralba,L.G. Franquelo, A. Fernández-León, F. Tortosa-López, D. Gonzáles-Gutiérrez, "A Hardware Approach for SEU Immunity using Xilinx FPGA's", XIX Conf. on Design of Circuits and Integrated System (DCIS'04), Bordeaux, French, November 2004.

[Kafka2006] Leoš Kafka, Ondřej Novák, "FPGA-based Fault Simulator", 9 Th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS'06), Prague, République Tchèque, pp. 218-219, Avril 2006

[Leveugle 1999] R. Leveugle, "Towards modeling for dependability of complex integrated circuits", 5thIEEE Int. On-Line Testing Workshop (IOLTW'99), Rhodes, Greece, pp. 194-198, July 1999.

[Civera 2001a] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, M. Violante, "Exploiting FPGA for accelerating fault injection experiments", Proc. 7th Int. On Line Testing Workshop (IOLTW'01), Taormina, Italy, pp. 9-13, July 2001.

[Civera 2001b] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, M. Violante, "FPGAbased Fault Injection for Microprocessor Systems", 10 Th Asian Test Symp. (ATS'01), Kyoto, Japon, p. 3004, November 2001.

[Civera2003] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, M. Violante, "New Techniques for Efficiently Assessing Reliability of SoCs", Microelectronics Journal, vol 34, issue 1, pp. 53-61, January 2003.

[Portela2004] M. Portela-Garcia, C. lópez-Ongil, M. García-Valderas, L. Entrena-Arrontes, "Analysis of Transient Fault Emulation Techniques in Platform FPGAs", XIX Conf. on Design of Circuits and Integrated System (DCIS'04), Bordeaux, French,November2004

[**López2005**] C. López-Ongil, M. García-Valderas, M. Portela-García, L. Entrena-Arrontes, "Autonomous Transient Fault Emulation on FPGAs for Accelerating Fault Grading", 11th IEEE Int. On-Line Testing Symposium (IOLTS'05), Saint-Raphael, French, pp. 43-45, July 2005.

[Garcia2006] M. Garcia-Valderas, M. Portela-García, C. López-Ongil, L. Entrena, "An Extension of Transient Fault Emulation Techniques to Circuits with Embedded Memories", The IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS'06), Prague, pp. 218-219, Avril 2006

[Laprie2004] J.-C. Laprie, "Sûreté de fonctionnement informatique: concepts, défis, directions", ACI Sécurité et Informatique, Toulouse, Novembre 2004.

[Johnson1989] Johnson, Barry W. "Design and Analysis of Fault Tolerant Digital Systems." Design and Analysis of Fault Tolerant Digital Systems.1989.

[Avizienis2004] A. Avizienis, J.C. Laprie. "Basic Concepts and Taxonomy of Dependable and Secure Computing." IEEE Transactions on Dependable and Secure Computing Archive, 2004: vol. 1.

[Arlat1992] Arlat, J. "Fault Injection for the Experimental Validation of Fault Tolerant Systems." Proceedings of Workshop Fault-Tolerant Systems. Kyoto, Japan: IEICE, 1992. 3340.

[Barbosa2012] Raul Barbosa, and Mario Zenha-Rela. 2013 Resilience Assessment and Evaluation of Computing Systems 2012, pp 263-281, Springer 2012

[Madeira2000] Madeira, H, D Costa, and M Vieira. "On the emulation of software faults by software fault injection." Proceedings International Conference on Dependable Systems and Networks. New York, NY: IEEE, 2000. 417-426.

[Maia2005] Maia, R, L Henriques, R Barbosa, D Costa, and H Madeira. "Xception fault injection and robustness testing framework: a case-study of testing." 2005.

[George2010(b)] N. George, C. Elks, B. Johnson, J. Lach. "Spatial Multi-Bit Soft-Error Tolerance in Logic." Dependable Systems and Networks Symposium. Chicago, IL, 2010.

[Gilles2011Th] « Simulation de fautes pour l'évaluation du test en ligne de systèmes RFID », doctoral dissertation, University of Grenoble, French 2011.

[Andres2008] D. D. Andres, J.C. Ruiz, D. Gil, P. Gil. "Fault Emulation for Dependability Evaluation of VLSI Systems." IEEE Transactions on Very Large Scale Integration (VLSI)Systems, 2008.

[Andres2006] D.D. Andres, J.C. Ruiz, D. Gil, P. Gil. "Run-Time Reconfiguration for Emulating Transient Faults in VLSI Systems." International Conference on Dependable Systems and Networks.2006. 291-300.

[Stott2000] D.T. Stott, B. Floering, D. Burke, Z. Kalbarczyk, R.K. Iyer. "Nftape: A Framework for Assessing Dependability in Distributed Systems with Lightweight Fault Injectors." IEEE, 2000.

[Vargas 2005] F. Vargas, D.L. Cavalcante, E. Gatti, D. Prestes, D. Lupi. "On the Proposition of an EMI-Based Fault Injection Approach." 11th IEEE International On-Line Testing Symposium (IOLTS 2005).Stevenson, Washington: IEEE, 2005. 207-20

[KARL1991] J. Karlsson, U. Gunneflo, P. Lidén, J. Torin "Two Fault Injection Techniques to Test of Fault Handling Mechanisms", Proc. of IEEE Int. Test Conference (ITC'91), pp. 140-149, 1991.

[KARL1995] J. Karlsson, P. Folkesson, J. Arlat, Y. Crouzet, G. Leber, "Comparison And Integration Of Three Diverse Physical Fault Injection Techniques", Predictably Dependable Computing Systems, pp. 309-327, 1995.

[Selmane2008] N. Selmane, S. Guilley, and J.-L. Danger, "Practical setup time violation attacks on AES," in Proceedings of the 7th European Dependable Computing Conference. IEEE, 2008, pp. 91–96.

[Fukunaga2009] T. Fukunaga and J. Takahashi, "Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers," in Proceedings of FDTC 2009. IEEE, 2009, pp. 84–92.

[Agoyan2010]M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria, "When clocks fail: On critical paths and clock faults," in Smart Card Research and Advanced Application –CARDIS 2010, ser. LNCS, D. Gollmann,J-L Lanet, and J. IguchiCartigny, Eds., vol. 6035. Springer-Verlag, 2010, pp. 182–193.

[Choi1992] G.S. Choi, R.K. Iyer. "Focus: An Experimental Environment for Fault Sensitivity Analysis." IEEE Transactions on Computers, 1992: 1515-1526.

[Jenn1994] 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.Pasadena, CA, 1994. 66-75.

[DeLong1996] DeLong, T A, B W Johnson, and J A Profeta. "A Fault Injection Technique for VHDL Behavioral-Level Models." 1996. 24-33.

[Kwang-Ting1999] C. Kwang-Ting, H. Shi-Yu, d. Wei-Jin. "Fault Emulation: A New Methodology for Fault Grading." IEEE Transactions on ComputerAided Design of Integrated Circuits and Systems, 1999: 1487-1495.

[Mezzah2005] Mezzah .Ibrahim 2005 "Study and design of microcontroller in FPGA" Setif university Algeria.

[Lala2012] P. Lala, "Transient and Permanent Fault Injection in VHDL Description of Digital Circuits," Circuits and Systems, Vol. 3 No. 2, 2012, pp. 192-199.

[Lyon1962] R. E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. IBM Journal, pages 200{209, Apr. 1962

[Boulé2005] M. Boulé, and Z. Zilic, "Incorporating Efficient Assertion Checkers into Hardware Emulation," IEEE Intl. Conference on Computer Design (ICCD'05), pp. 221-228, 2005.

[Accellera2013] "Accellera Standard OVL V2 Library Reference Manual," Accellera Systems Initiative Inc., CA, USA, January 2013, accellera.org.

[TAO2009] Y. Tao, "An introduction to assertion-based verification," in IEEE 8th International Conference on ASIC 2009 (ASICON'09), October 2009, IEEE, pp. 1318-1323.

[Miremadi1995] G. Miremadi, J. Torin. "Evaluating Processor-Behavior and Three Error Detection Mechanisms Using Physical Fault-Injection." IEEE Transactions on Reliability, 1995: 441-454.

[Lenhart2007]Lenhart, T. A Formalized Verification Methodology for Soft IP Cores Used in Safety-Critical Hardware Applications. Charlottesville, VA: University of Virginia, M.S. Thesis, 2007.

[Boume2007] M. Boule, J.-S. Chenard, and Z. Zilic, “Assertion checkers in verification, silicon debug and in-field diagnosis,” In 8th International Symposium on Quality Electronic Design 2007, pp. 613-620, Washington, DC, USA, IEEE, 2007.

[Riazati2006] M. Riazati, S. Mohammadi, A. Afzali-Kusha and Z. Navabi, “Improved Assertion Lifetime via Assertion-Based Testing Methodology,” in International Conference on Microelectronics (ICM’06), IEEE, 2006, pp. 48-51.

[Kubalik2006] P. Kubalik, P. Fiser, and H. Kubatova, “Fault tolerant system design method based on self-checking circuits,” in 12th IEEE InternationalOn-Line Testing Symposium, (IOLTS 2006), IEEE, 2006, pp. 2-pp.

[Boule2005] M. Boule, and Z.Zilic, “Incorporating efficient assertion checkers into hardware emulation,” in International Conference on Computer Design 2005, (ICCD’05), IEEE, pp. 221-228.

[Abarbanel2000]Y. Abarbanel, I. Beer, L. Gluhovsky, S. Keidar, and Y. Wolfsthal, “Focs–automatic generation of simulation checkers from formal specifications,” in 12th International Conference onComputer Aided Verification, 2000, pp. 538-542, Springer Berlin Heidelberg.

[Kakoee2008] M. R. Kakoee, M. Riazati, And S. Mohammadi, “Enhancing the testability of RTL designs using efficiently synthesized assertions,” in 9th International Symposium on Quality Electronic Design (ISQED 2008), IEEE, 2008, pp. 230-235.

[Sampson-97] J.R. Sampson, W. Moreno, F. Falquez, "Validating Fault Tolerant Designs using Laser Fault Injection (LFI)", IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems(DFT’97), Paris, France, pp. 175-183, Octobre 1997.

[Duzellier-00] S. Duzellier, D. Falguère, L. Guibert, V. Pouget, P. Fouillat, R. Ecoffet, "Application ofLaser Testing in Study of SEE Mechanisms in 16-Mbit DRAMs", IEEE Trans. On Nuclear Science, vol. 47, n° 6, Décembre 2000.

[LEWIS-05],D. Lewis, V. Pouget, F. Beaudoin, G. Haller, P. Perdu, P. Fouillat, “Implementing Laser-Based Failure Analysis Methodologies Using Test Vehicles”, IEEE Trans. on Semiconductor Manufacturing, vol. 18, n° 2, Mai 2005.

[Civera2001]P. Civera, L. Macchiarulo, M. Rebaudengo, M. SonzaReorda, M. Violante, "Exploiting FPGA for accelerating fault injection experiments", Proc. 7th Int. On Line Testing Workshop (IOLTW’01), Taormina, Italie, pp. 9-13, Juillet 2001

[Seward2003] S. R. Seward, and P. K. Lala, “Fault injection for verifying testability at the VHDL level,” in International Test Conference, IEEE, 2003, pp. 131-137.

[Patel1982] J. H. Patel and L. Y. Fung. Concurrent Error Detection in ALU’s by Recomputing with Shifted Operands. IEEE Transactions on Computers, 31(7) :589–595,1982

[Pfanz2002] M. Pflanz, K. Walther, C. Galke and H. T. Vierhaus, "On-line error detection and correction in storage elements with cross-parity check," On-Line Testing Workshop, 2002. Proceedings of the Eighth IEEE International, 2002, pp. 69-73.

[Douglas2002] Douglas L. Perry, "VHDL: programming by example", McGraw-Hill, 4eedition, 2002

[AMS2014]http://ams.com/eng/Support/Demoboards/Wireless Connectivity/Readers/AS3992-Demo-Kit-

[Banga2007] M. Banga and M.S. Hsiao, “A Region Based Approach for the Identification of Hardware Trojans”, HOST, 2008.

[Jin2009]Y. Jin, N. Kupp and Y. Makris. Experiences in Hardware Trojan Design and Implementation. 2009

[Chakraborty2009] R. S. Chakraborty, S. Narasimhan and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," High Level Design Validation and Test Workshop, 2009. HLDVT 2009. IEEE International, San Francisco, CA, 2009, pp. 166-171

[Banga2008] M. Banga and M.S. Hsiao, “A Region Based Approach for the Identification of Hardware Trojans”, HOST, 2008.

[Chen2008] Z. Chen et al, “Hardware Trojan Designs on BASYS FPGA Board (Virginia Tech)”, CSAW Embedded System Challenge, 2008. [Online]. Available: http://isis.poly.edu/~vikram/vt.pdf

[HeLi 2016] He Li, Qiang Liu, Jiliang Zhang, A survey of hardware Trojan threat and defense, Integration, the VLSI Journal, February 2016, ISSN 0167-9260.

[waksman2011]Waksman, A. & Sethumadhavan, S. (2011) Silencing hardware backdoors, in Proceedings of the 32nd IEEE Symposium on Security and Privacy, May 2011.

[Bhunia2010]Chakraborty, R. S., Narasimhan, S. & Bhunia, S. (2010) Hardware trojan: Threats and emerging solutions. http://www.trust-hub.org/resources/113.

[WANG2008]Wang, X., Tehranipoor, M. & Plusquellic, J. Detecting malicious inclusions in secure hardware: Challenges and solutions, IEEE International Workshop on HardwareOriented Security and Trust 0, 15–19.

[Karri2010] R. Karri et al., "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," Computer, Oct. 2010, pp. 39-46.

[Jin2010]Y. Jin and Y. Makris. Hardware Trojans in Wireless Cryptographic ICs. Design Test of Computers, IEEE, 27(1):26 {35, jan. 2010

[WOLF2008]Wolff, F., Papachristou, C., Bhunia, S. & Chakraborty, R. S. (2008) Towards trojan-free trusted ics: problem analysis and detection scheme, in Proceedings of the conference on Design, automation and test in Europe, DATE '08, ACM, New York, NY, USA, pp. 1362 136.

[Mark2011]Mark Beaumont, Bradley Hopkins, and Tristan Newby. Hardware trojans-prevention, detection, countermeasures (a literature review). Technical report, DTIC Document, 2011.

[Chakraborty2008]Chakraborty, R. S., Paul, S. & Bhunia, S. (2008) On-demand transparency for improving hardware trojan detectability, in IEEE International Symposium on HardwareOriented Security and Trust.

[Salmani2009] Salmani, H., Tehranipoor, M. & Plusquellic, J. (2009) New design strategy for improving hardware trojan detection and reducing trojan activation time, in Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on, pp. 66 –73.

[Jha2008] Jha, S. & Jha, S. K. (2008) Randomization based probabilistic approach to detect trojan circuits, High-Assurance Systems Engineering, IEEE International Symposium on0, 117–124.

[Abramovici2009] Abramovici, M. & Bradley, P. (2009) Integrated circuit security: new threats and solutions, in Proceedings of the 5th Annual Workshop on Cyber Security and Information intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies, CSIIRW '09, ACM, New York, NY, USA, pp. 55:1–55:3.

[Wolff2008] F Wolff, C. Papachristou, S. Bhunia and R. S. Chakraborty. Towards Trojan-Free Tusted ICs: Problem Analysis and Detection Scheme. 2008

[Chakraborty2009] R. S. Chakraborty and S. Bhnia. Security against Hardware Trojan through a Novel Application of Design Obfuscation.ICCAD'09. 2009

[Fritz2010] G. Fritz, V. Beroulle, O. Aktouf, M. D. Nguyen, D. Hély, "RFID System On-line Testing Based on the Evaluation of the tags Read-Error-Rate", Journal of Electronic Testing, pp 1-10, december 2010.

[Floerkemeier2005] Floerkemeier, C. and Lampe, M. RFID middleware design -addressing application requirements and RFID constraints. InsOc-EUSAI 05, Oct. 2005.

[Welbourne2007] Welbourne, E. Balazinska, M. Borriello, G. Brunette, W. "Challenges for Pervasive RFID-Based Infrastructures," Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on , vol., no., pp.388,394, 19-23 March 2007

[Mitrokotsa 2010]Mitrokotsa, A., M. Rieback, and A. Tanenbaum, "Classifying RFID attacks and defenses," Information Systems Frontiers, vol. 12, pp. 491-505, 2010.

[Tehranipoor2011] M. Tehranipoor, H. Salmani, X. Zhang, X. Wang, R. Karri, J.Rajendran, and K. Rosenfeld, Trustworthy Hardware: Trojan Detection and Design-for-Trust Challenges, Computer Magazine, July 2011

[Tehranipoor2010] M. Tehranipoor and F. Koushanfar. A Survey of Hardware Trojan Taxonomy and Detection. IEEE Design & Computers. 2010

[Tehranipoor2012] S. Skorobogatov and C. Woods, "Breakthrough Silicon Scanning Discovers Backdoor in Military Chip",Cryptographic Hardware and Embedded Systems – CHES 2012, Lecture Notes in Computer Science Volume 7428, 2012, pp 23-40

[EPC2008] EPCglobal, EPC radio frequency idenrity protocols classe-1 generation-2 UHF RFID, protocol for communications at 860 MHz 960 MHz, version 1.0.9, 2004

[Bhnia2009] R. S. Chakraborty and S. Bhnia. Security against Hardware Trojan through a Novel Apllication of Design Obfuscation. ICCAD'09. 2009

[Bhnia2014] S. Bhunia, M. S. Hsiao, M. Banga and S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures," in *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229-1247, Aug. 2014.

[Rieback 2006] Rieback, M.R.; Crispo, B.; Tanenbaum, A.S., "RFID malware: truth vs. myth," Security & Privacy, IEEE , vol.4, no.4, pp.70,72, July-Aug. 2006

[King 2008] S.T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, Y. Zhou, Designing and implementing malicious hardware, Leet (2008).

[Lin2008] L. Lin, M. Kasper, T. Gneysu, C. Paar, W. Burleson, Trojan side-channels: lightweight hardware Trojans through side-channel engineering, In: Lecture Notes in Computer Science, vol. 5747, 2009, pp. 382-395.

[Rajendran2010], J., Gavas, E., Jimenez, J., Padman, V. & Karri, R. (2010) Towards a comprehensive and systematic classification of hardware trojans, in Circuits and Systems(ISCAS), Proceedings of 2010 IEEE International Symposium on, pp. 1871 –1874

[Omar2013] Ibrahim Mezzah, Omar ABDELMALEK, Vincent Beroulle, David Hély"Assertion Based On-line Fault Detection Applied on UHF RFID tag" 8TH IEEE INTERNATIONAL DESIGN & TEST SYMPOSIUM (IDT 2013).

[Banga2009]Banga, M. & Hsiao, M. S. (2009) A novel sustained vector technique for the detection Abstraction, The 22nd International Conference on VLSI Design, New Delhi, India, 5-9 January 2009, IEEE, pp. 327–332 of hardware trojans, in VLSI Design 2009

Publications

International Conference

1) **Omar Abdelmalek**, D Hély, V Beroulle "EPC Class 1 GEN 2 UHF RFID Tag Emulator for Robustness Evaluation and Improvement" in Proceedings of 8th International Conference on Design &Technology of Integrated Systems in Nanoscale Era (DTIS),Abu Dhabi,2013

2) **Omar Abdelmalek**, David Hely, and Vincent Beroulle, Ibrahim Mezzah "An UHF RFID Emulation Platform with Fault Injection and real time Monitoring capabilities" 8TH IEEE INTERNATIONAL DESIGN & TEST SYMPOSIUM (IDT 2013).

3) Ibrahim Mezzah, **Omar Abdelmalek**, Vincent Beroulle, David Hély "Assertion Based On-line Fault Detection Applied on UHF RFID Tag" 8TH IEEE INTERNATIONAL DESIGN & TEST SYMPOSIUM (IDT 2013).

4) Eduardo Hidalgo, **Omar Abdelmalek**, David Hély, Vincent Beroulle "Triggering Hardware Trojans in EPC C1G2 RFID Tags", First Workshop on Trustworthy Manufacturing and Utilization of Secure Devices,(trudevice2013), Avignon, France.

5) **Omar Abdelmalek**, Vincent Beroulle, David Hély,"FaultTolerance Evaluation of RFID Tags", 15th IEEE Latin-American Test Workshop, (LATW2014) Fortaleza, Brazil.

6) **Omar Abdelmalek**, Vincent Beroulle, David Hély, "Emulation of Fault Injection on UHF RFID Transponder" 17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems,(DDECS 2014) Warsaw, Poland,

National Conference

Omar Abdelmalek, Vincent Beroulle, David Hély, Design and emulation of new robust architectures for UHF RFID Tags in Colloque National du GDR SoC-SiP, Paris 2012