AN ARCHITECTURAL APPROACH FOR REDUCING POWER AND INCREASING SECURITY OF RFID TAGS

by

Shen-Chih Tung

B.S. National Taiwan Ocean University, Taiwan, 1997

M.S. Telecommunication, University of Pittsburgh, 2000

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2008

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Shen-Chih Tung

It was defended on

March 28, 2008

and approved by

Alex K. Jones, Assistant Professor, Department of Electrical and Computer Engineering

Marlin H. Mickle, Nickolas A. DeCecco Professor, Electrical Engineering Department

James T. Cain, Professor Emeritus, Department of Electrical Engineering

Allen Cheng, Assistant Professor, Department of Electrical Engineering

Ahmed Amer, Assistant Professor, Department of Computer Science

Dissertation Director: Alex K. Jones, Assistant Professor, Department of Electrical and Computer Engineering

AN ARCHITECTURAL APPROACH FOR REDUCING POWER AND INCREASING SECURITY OF RFID TAGS

Shen-Chih Tung, PhD

University of Pittsburgh, 2008

Radio Frequency Identification (RFID) technology is currently employed for a variety of applications such as RFID-based wireless payment, healthcare, homeland security, asset management, etc. Due to newer privacy requirements and increasingly secure applications, typical RFID tags are required to expand security features such as data encryption and safe transactions. However, RFID tags have extremely strict low-power consumption requirements. Thus, reduced power consumption and secure data transactions are two main problems for the next generation RFID tags. This dissertation presents an architectural approach to address these two main problems.

This dissertation provides a multi-domain solution to improve the power consumption and security, while also reducing design time and verification time of the system. In particular, I describe (1) a smart buffering technique to allow a tag to remain in a standby mode until addressed, (2) a multi-layer, low-power technique that transcends the passive-transaction, physical, and data layers to provide secure transactions, (3) an FPGA-based traffic profiler system to generate traces of RFID communications for both tag verification and power analysis without the need of actual hardware, and (4) a design automation technique to create physical layer encoding and decoding blocks in hardware suitable for RFID tags.

This dissertation presents four contributions: (1) as a result, based on a Markov Process energy model, the smart buffering technique is shown to reduce power consumption by 85% over a traditional active tag; (2) the multi-layer, low-power security technique provides protection against malicious reader attacks to disable the tag, to steal the information stored in or communicated to the device. The power consumption overhead for implementing these layers of security is increased

approximately 13% over the basic tag controller; (3) in addition, the FPGA-based traffic profiler system has been able to generate traces for ISO 18000 part 6C (EPC Gen2) protocol; and (4) the designs of encoding/decoding blocks are generated automatically by the Physical Layer Synthesis tool for five protocols used in or related to RFID. Consequently, any power consumption of five designs is less than 5 μW. Furthermore, compared with five designs implemented by hand, the difference of the power consumption between two of them is less than 7% at most.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

This dissertation is dedicated to my parent, who gave me an utmost support and encouragement. Their spiritual support helps me overcome every difficulty.

Importantly, I would like to express my deep gratitude for my advisor Dr. Alex K. Jones, who supported me with his guidance and his encouragement. He is an excellent mentor and researcher. I am very impressed his intelligence. His creative idea and insightful thought guide me to finish my research. Without his guidance, support and help, it is impossible to accomplish this dissertation.

I would like to express my greatly appreciation to Professors Marlin Mickle and Tom Cain for providing supports and recommendations. Without their support, I will not be able to continue my research in University of Pittsburgh. Their recommendations and suggestions are invaluable for my research. In addition, I would like to thank Professor Allen Cheng and Professor Ahmed Amer for their guidance.

I would like to thank Dr. Ray Hoare from the bottom of my heart. He led me, taught me, supported me, and helped me. He is not only a teacher to me, but also a great friend.

I would like to express my special thanks to my colleagues in our EDA labs: Swapna Dontharaju, Colin Ihrig, Justin Stander, Yu Zhang, Gerold Dhanabalan, and Gayatri Mehta.

Words alone cannot express my thanks I owe my wonderful and lovely wife, Ying Liu, and my lovely son, Daniel. I am deeply grateful to my wife for her love, sacrifice and support. Daniel's smile and hug strongly support me. I am so proud of them.

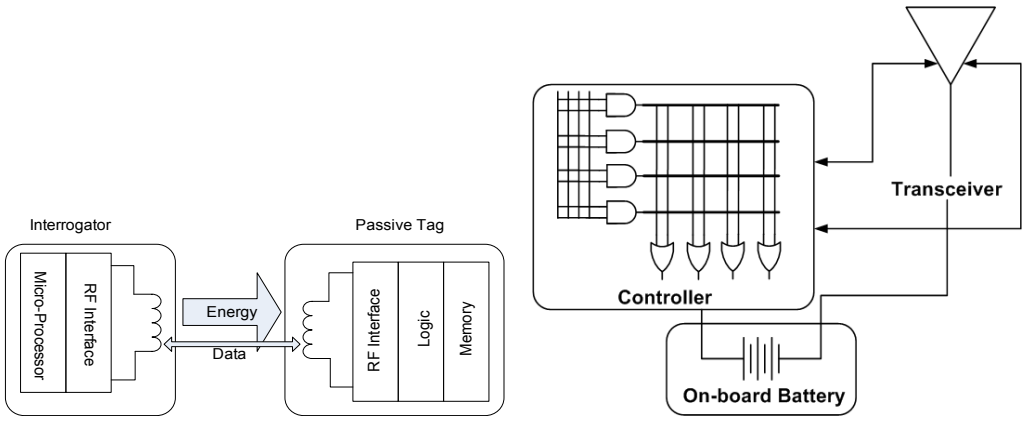
Finally, I would like to thank God. He gives me everything.

1.0 INTRODUCTION

Radio Frequency IDentification (RFID) technology is an automatic identification technology that uses radio waves carrying energy or data between interrogators or unidentified tags/objects. An RFID system contains two main components: interrogators and RFID tags. Active and passive are two dominant types of RFID tags shown in Figure 1. A passive RFID tag is also known as a passive transponder including a transponder antenna, RF interface, logic circuitry and memory shown in Figure 1(a) [1, 2]. It does not have an internal power supply for the logic circuitry and memory. The energy powering the entire passive tag is provided by an interrogator. Therefore, the restriction of the power supply limits the transmission range of a passive RFID tag and the complexity of the tag operation.

Unlike a passive RFID tag, an active RFID tag has an internal power supply, usually a battery. In general, a traditional active RFID tag is composed of an RF transceiver, a processor-based controller, and an on-board battery as shown in Figure 1(b). Since an active RFID tag is powered by an internal power supply, it supports more complex tag operations and transmit responded signals with a long distance (around 300 feet). However, the tag's life relies on the battery life. Therefore, the low power consumption is a main issue for the user to design and develop active RFID systems.

Active RFID technology confronts several problems in energy dissipation, security, system verification, and design cost. Section 1.1 addresses the motivation of four problems. First of all, the waste of power dissipation is discovered in a traditional active RFID tag. If the power dissipation can be reduced, a battery life can be extended or an active tag can perform advanced operations, such as advanced security operations, without diminishing the battery life. More detailed power dissipation is described in Section 1.1.1.



(a) Passive RFID tag. (b) Active RFID tag.

Figure 1: An architectural overview of RFID tags.

The second problem is the need of low-power security features. Traditional RFID tags provide security features by activating their processor-based controller with increasing power consumption. Thus, this is a fundamental trade-off situation: the need for stronger security features is increasing complexity and energy consumption of RFID tags. However, RFID systems require security features to be implemented using techniques that provide a high degree of protection while not significantly increasing the complexity of the system as complexity can increase power consumption and implementation cost. Thus, it is important to develop new security techniques that take advantage of the fundamental properties of RFID communication such as physical layer protocols, low-power communication extensions, sleep modes, physical implementation variations, etc. More detailed information is described in Section 1.1.2.

Thirdly, RFID technology requires a verification scheme for the network traffic of RFID systems. A computer network debugging tool, *tcpdump* as an example, allows the user to intercept TCP/IP packets and display them. Upon the collected packet information, the user is able to verify and analyze the network properties, monitoring the traffic, and even providing a protection to the network. However, there is no tool like *tcpdump* in RFID systems, which provides detailed network traffic of RFID communications from the system level perspective. This concept is described in Section 1.1.3.

The fourth problem of RFID technology is the increasing design cost for designing and implementing a variety of RFID systems. In order to meet the requirements for different RFID specifications and support extensive RFID applications, multiple standards exist. In most application, RFID tag and reader hardware software must be specifically designed for each applications. This keeps overall design time long and the system costs high. Section 1.1.4 presents the impacts of the design automation technology on RFID systems.

This dissertation provides a multi-domain solution to improve the power consumption and security, while also reducing design time and verification time of RFID systems. In particular, this dissertation describes:

1. Smart buffering technique to allow a tag to remain in a standby mode until addressed. This is the solution for reducing power consumption of active RFID tags.
2. A multi-layer, low-power technique that transcends the passive-transaction, physical, and data layers to provide secure transactions. This technique provides high degree protection for active RFID tags with minimum power consumption.
3. An FPGA-based traffic profiler system to generate traces of RFID communications for both tag verification and power analysis without the need for actual hardware. This technique is a system-level verification platform for RFID systems.
4. A design automation technique to create physical layer encoding and decoding blocks in hardware suitable for RFID tags. This technique is a solution for reducing design cost and time.

The remainder of this chapter is organized as follows: Section 1.1 presents the motivation of each contribution including problem statement in more detail. Section 1.2 presents the dissertation statement including a concise statement of the contributions of this work. The impact of the contributions is described in Section 1.3. The outline of the dissertation is listed in Section 1.4.

1.1 MOTIVATION

1.1.1 The Energy Dissipation of Active RFID Systems

A transceiver and a processor-based controller are two dominating components of a traditional active RFID tag shown in Figure 1(b). Two significant sources of power dissipation can be observed during the operation of active RFID tags, in terms of the power consumption of the transceiver and of the processor-based controller. One of the significant sources of power consumption is the need to energize the transceiver all the time in order to receive any incoming signal. Since the RFID interrogator dominates the communication, every communication will be initiated by the reader. The energy consumed by a receiver (E_r) is proportional to the time (T_r)when the receiver is energized, if the power consumption of the receiver (P_r) is constant, shown in Eq. (1.1). The key to reducing the energy consumed in the receiver (E_r) is to compress T_r.

$$\begin{aligned} E_r &= E_r^s + E_r^a \\ &= P_r^s \cdot T_r^s + P_r^a \cdot T_r^a \\ K &= T_r^s + T_r^a \end{aligned} \tag{1.1}$$

$$\begin{aligned} E_c &= E_c^s + E_c^a \\ &= P_c^s \cdot T_c^s + P_c^a \cdot T_c^a \\ &= P_c^s \cdot T_c^s + P_c^a \cdot (T_{sc} \cdot N_c) \\ K &= T_c^s + T_c^a \end{aligned} \tag{1.2}$$

The other significant portion of power dissipation derives from the processor-based microcontroller. A microcontroller needs 2.2 to 8 *mA* to stay in the active mode and ONLY needs 1 to 15 μA to stay in the sleep mode. However, in RFID systems, the microcontroller of an active tag is awakened by those incoming commands which are intended for other tags.

Broadcasting is the only communication method between RFID readers and tags. However, RFID technology uses not only one-to-all RFID commands but also point-to-point commands

specified in ISO 18000 Part 7 [3] and ANSI 256 [4] active RFID standards. `Collection` or `Collection with Data` in ISO 18000 Part 7 are two common one-to-all commands. An RFID reader broadcasts this type of commands to all RFID tags and they wake up and respond associated information back to the reader.

For point-to-point commands such as `Sleep` or `User ID` from ISO 18000-7, the RFID reader broadcasts a point-to-point command to all RFID tags even though it attempts to access only one tag. All tags wake up and process the incoming command. However, only one of tags needs to respond with the associated information while the rest of the tags ignore the incoming command. For those tags ignoring the incoming command, on-tag controllers dissipate energy for receiving and dropping commands which could be avoided.

Eq. (1.2) shows the energy consumed by a processor-based controller for an active RFID tag. The total energy dissipated is the summation of E_c^s and E_c^a. E_c^s presents the energy consumed during the controller in the sleep/standby mode, which can achieved by the power consumption (P_c^s) times the duration (T_C^s). And $E_c^a = P_c^a \times T_c^a$ presents the energy consumed by a controller when it is activated, where P_c^a illustrates the power consumption when a controller is in the activation mode and T_c^a indicates the duration when a controller processing commands. The time variable, T_c^a, is determined by the number of processed commands (N_c) and the time of processing a single command (T_{sc}).

The total energy consumed by a processor-based controller (E_c) is proportional to the number of commands (N_c) processed in it shown in Eq. (1.2). Based on the point-to-point communication model, Figure 2 illustrates the impact of the reduction of the processed commands (N_c) on the energy consumed by a microcontroller in an active RFID tag. The solid line shows the energy consumed for one active tag when the number of total tags in valid communication range is increased to 20 tags.

The dashed line and the dotted line in Figure 2 represent the idea that assumes the number of commands (N_c) processed in a tag is reduced by 50% and 75% perspectively. The dashed line predicts the result of energy dissipation when the number of processed commands is reduced by 50%. Furthermore, the dotted line indicates the energy saving result when N_c is reduced by 75%. Based on general assumption, these preliminary results provide a research tendency and focus; however, they are not accurate for predicting how much power can be conserved. We need a

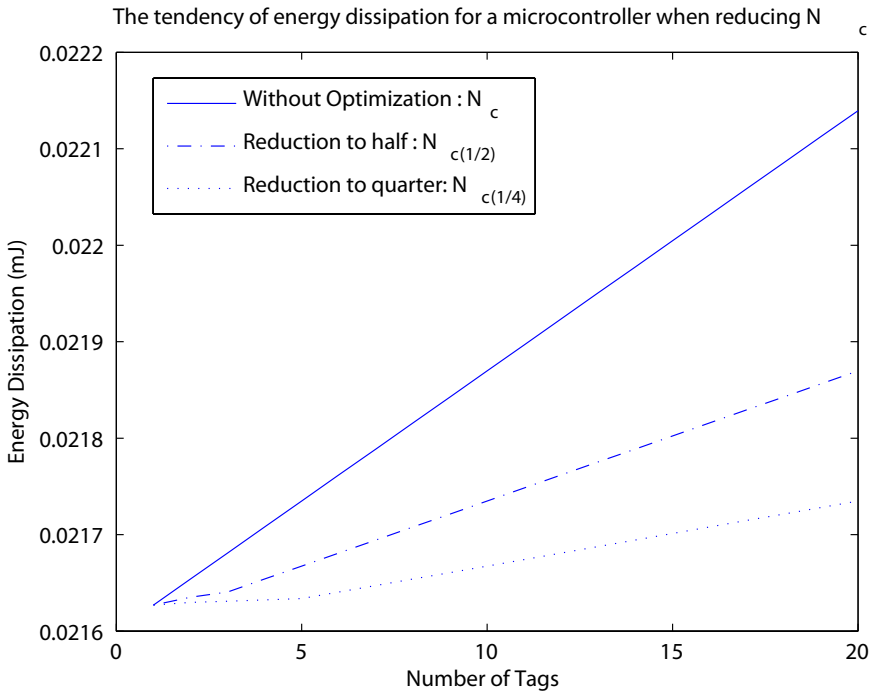


Figure 2: The preliminary results illustrating the tendency of energy dissipation for a microcontroller.

solution that minimizes the time when the controller is active to save power.

1.1.2 Security and Energy Trade-off

Traditional RFID tags provide security features by activating their processor-based controller with increasing power consumption. This situation creates a security strength versus energy trade-off. However, RFID systems require security features to be implemented using techniques that provide a high degree of protection while not significantly increasing the complexity of the system. Complexity can increase power consumption and implementation cost. Thus, it is important to develop new security techniques that take advantage of the fundamental properties of RFID communication such as physical layer protocols, low-power communication extensions, sleep modes, physical implementation variations, etc.

Application		Animal Tagging	Inventory Management	Door Access	Transit Payment	Credit Card Payment	Secure Employee ID	ePassport
Standard Used	ISO 14443			X	X	X	X	X
	ISO 15693		X	X				
	ISO/IEC 18000-7		X					
	ISO 11784	X						
Application Requirements	Operation Range	Medium	Short to long	Short to Medium	Short	Short	Short	Short
	Stored Information	ID No.	Product Code	ID No.	Fare value	Account Info.	Employee ID No.	Personal Info.
	Security level	Low	Low to High	Low to High	Medium to High	High	High	High

Figure 3: The security requirements and standards for RFID applications.

Applications for RFID continue to expand into domains such as electronic passports, electronic payment systems, and electronic container seals. These applications have a risk of unauthorized access to sensitive biometric or financial information through the RFID tag or tag communication. Figure 3 illustrates a variety of RFID applications with associated applied standards and security requirements [5].

The inventory management application requests security requirements from low-level to high-level. For example, if the RFID technology is applied to an asset or stock management, it may request a low-level sensitivity of security requirement. However, when containers containing confidential material or goods, the security level needs to be roused to high sensitivity. For instance, the Department of Defense (DoD) sees combining RFID technology with satellite communication as a new approach to improve the performance and security of the military's supply operations [6].

The ePassport is one of the RFID applications requiring high-level security protection. For instance, the Department of Homeland Security sees RFID technology as a means to improve

Table 1: Power requirements of microcontrollers.

Model	PIC18F6720	MSP430F16x	ATmega128L
Frequency (MHz)	20	8	8
Word size (bit)	8	16	8
Power (awake; mA)	2.2	2	8
Power (Sleep; μA)	1	1.1	15

the ability to match entries without significantly increasing processing time or invading visitors' privacy [7].

However, as RFID devices are intended to be small and relatively simple devices, security protocols and techniques can significantly lag behind the other details such as correctness, read rate, power consumption, etc. As such, the state of security in RFID systems is generally weak compared to other mature computational technologies such as Internet servers, shared computing workstations, and even smartcards. RFID systems in general and RFID tags in particular have the requirement of a power versus area tradeoff. The power consumed by the tags impacts the range of the device (passive) or lifetime of the tag (active), where the area dictates the cost of the device. In general, the industry has focused more on the area constraint than the power constraint for passive tags. The 5 cent RFID tag is a famous example of an area optimized device for reduction in cost. For instance, the 5 cent tag, one example of passive RFID tags, is composed of less than 10,000 gates [8].

Table 1 lists microcontrollers used for prototyping active RFID tags or commercial products in wireless sensor networks [9, 10, 11]. An active tag needs to provide adequate power, from $2mA$ to $8mA$, to provide a microcontroller enough energy to perform security features.

An Application-Specified Integrated Circuit (ASIC) technology has been proved that it consumes a small amount of energy to perform advanced security operations. For example, the Advanced Encryption Standard (AES), a strong encryption algorithm, is designed and implemented by using 0.18μ technology. This design consumes 8.15 μA with adequate 992 clock cycles and 307 μA with adequate 10 clock cycles [12].

Security features have often been a secondary concern for vendors who provide passive RFID tags primarily because strong authentication and encryption algorithms are complex and would significantly increase the cost and power budget of the tag. Sarma *et al.* discusses the security risks and challenges of low cost RFID tags [8]. For example, in the 5 cent tag, less than 1,000 gates are dedicated to security. In contrast, commercial implementations of the Advanced Encryption Standard require approximately 4,000 - 173,000 gates [12].

Unlike passive tags, expanding security features in active RFID tags may be affordable, but still impacts the lifetime of the tags. The lifetime of active RFID tags is related to fixed power sources (eg. on-board battery). As providing more security features, active RFID tags increase not only the complexity of tags, but also the power consumption. For example, an active tag performs a software-based data encryption algorithm. The processor-based controller needs to stay in active mode for a longer time to perform the data encryption. This tag consumes more energy than that does not provide the same security feature. Therefore, the active RFID systems needs a new technique that can provide not only high degree of protection, but also requiring a minimum power dissipation (eg. within the range of μW) to support these security features.

1.1.3 System-level Verification Platform for RFID Systems

Verification of RFID technology, especially the network traffic of RFID systems, is currently ad hoc requiring expensive equipment. In computer network debugging tools like *tcpdump* as an example allows the user to intercept TCP/IP packets and analyze them. Upon collecting the packet information, the user is able to verify and analyze the network, monitor the traffic, and even provide protection to the network. However, there is no tool like *tcpdump* in RFID systems, which provides detailed network traffic of RFID communications from the system level perspective.

Some theoretical traffic models, such as ALOHA traffic models [13, 14, 15],were developed to describe RFID communication patterns. However, those traffic models are not capable of providing precise views of RFID-command exchanges among interrogators and tags. To test new RFID systems or protocols, it is valuable to have real workloads from RFID deployments. In order to provide a precise real workloads or network traffic profile of RFID communication for power consumption analysis, a new verification platform is required to capture the RFID command traffic,

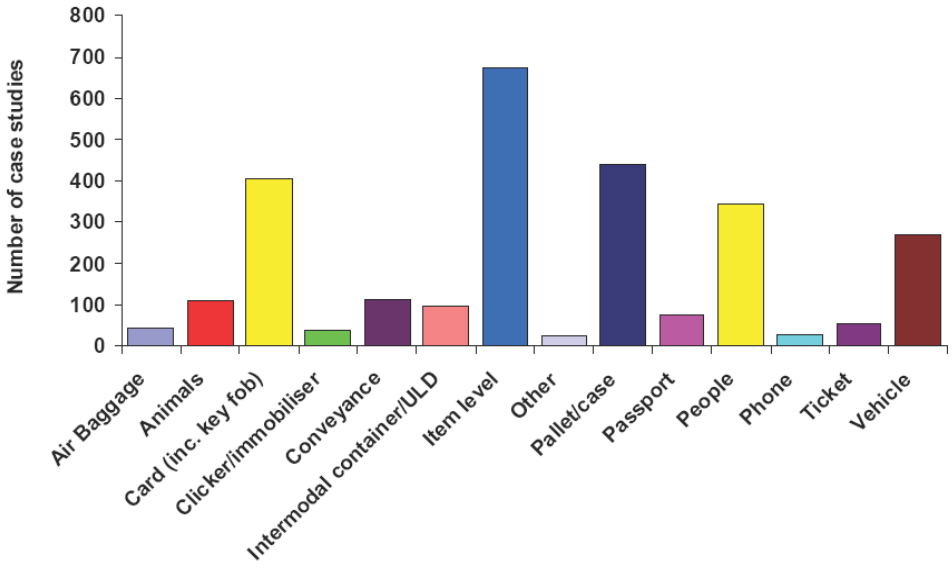


Figure 4: Case studies of RFID technology in 2008.

analyze RFID commands, and output the traffic profile to a readable text file.

1.1.4 The High Design Cost and Long Design Time for New RFID Applications

The main problem in developing new RFID systems or applications is the increasing design cost for designing and implementing these systems in a case by case basis. In order to meet the requirements for different RFID circumstances and support extensive RFID applications, multiple standards exist. In most applications, RFID tag and reader hardware software must be specifically designed for each applications. This keeps overall design time long and the system costs high. For example, RFID systems are expanding rapidly with their applications in a wide range of areas. These systems are used for a wide range of applications that track, monitor, report, and manage items as they move between different physical locations. A survey of RFID applications [16] depicts the summation of extensive new research including 3096 RFID projects in late 2007 shown in Figure 4. Based on this report, item-level management, contactless card, vehicle applications (e.g. E-Z Pass), and pallet flow control, are primary applications dominating the fields of academic research and RFID marketing.

A wide range of extensions such as memory, sensors, encryption, and access control can be added to the tag. The interrogators query the tags for information stored on them, which can include items like identification numbers, user written data, or sensory data shown in Figure 3. Tags contain a unique tag identification number and potentially additional information of interest to manufacturers, healthcare organizations, military organizations, logistics providers and retailers, or others that need to track the physical location of goods or equipment.

In order to meet the requirements for different RFID circumstances and support extensive RFID applications, multiple, often competing, standards exist (ISO=IEC JTC1, ANSI, EPC, etc.) for RFID hardware, software, and data management. Thus, most of the RFID systems are deployed for closed loop applications using either proprietary protocols or nonintersecting standards with nonreusable tags and readers.

As a result, in most applications, RFID tag and reader hardware and software must be specifically designed for each particular application, and must be physically modified or redesigned every time the specification for the current application is adjusted, as new applications are introduced, as the standards are modified or as new standards are developed. This situation keeps the overall design time long and the system costs high. It is important to develop a new design automation technique for RFID systems to reduce the design cost and design time.

1.2 DISSERTATION STATEMENT

This dissertation provides a multi-domain solution to improve the power consumption and security, while also reducing design time and verification time of RFID systems. This research is placed in the context of an innovative architecture for active RFID tags called Passive Active RFID Tag (PART), which integrates the benefits of passive and active RFID tags. PART is designed for a low-power, long-range and high-secure RFID solution described in Section 2.2.

In particular, I describe (1) a smart buffering technique to allow a tag to remain in a standby mode until addressed, (2) a multi-layer, low-power technique that transcends the passive-transaction, physical, and data layers to provide secure transactions for active RFID tags, (3) an FPGA-based traffic profiler system to generate traces of RFID communications for both tag ver-

ification and power analysis without the need for expensive verification tools, and (4) a design automation technique to create physical layer encoding and decoding blocks in hardware suitable for RFID tags.

This dissertation presents four contributions: (1) As a result, based on a Markov Process energy model, the smart buffering technique is shown to reduce power consumption by 85% over a traditional active tag; (2) The multi-layer, low-power security technique provides protection against malicious reader attacks to disable the tag, to steal the information stored in or communicated to the device. The power consumption overhead for implementing these layers of security is increased approximately 13% over the basic tag controller; (3) In addition, the FPGA-based traffic profiler system has been able to generate traces for ISO 18000 part 6C (EPC Gen2) protocol; and (4) The designs of encoding/decoding blocks are generated automatically by the Physical Layer Synthesis tool for five protocols used in or related to RFID. Consequently, any power consumption of five designs is less than 5 μW. Furthermore, compared with five designs implemented by hand, the difference of the power consumption between two of them is less than 7% at most.

1.3 IMPACT OF THE CONTRIBUTIONS

This dissertation presents a multi-domain solution to improve the power consumption and security, while also reducing design time and verification time of RFID systems.

After analyzing the operation of active RFID tags, we found that two significant sources of power dissipation can be saved while a processor-based controller remains in a sleep/standby mode. Based on the PART architecture, the *Smart Buffer* technique behaves like a guard, blocking any incoming commands carrying incorrect information. Therefore, the processor-based tag controller remains in a sleep/standby mode until a legitimate command arrived. The *Smart Buffer* technique is implemented by a dedicated hardware circuit which dissipates a small amount of energy compared with the processor-based controller. Based on our experimental result, it shows that using the *Smart Buffer* technique helps an active RFID tag to reduce the power consumption by 85%.

From security perspective, low-power security features are applied to different levels in the RFID tag communication structures from the physical layer through encryption of the data trans-

mission. *Multi-layer security* proposed in this dissertation provides protection against malicious reader attacks used to disable the tag, in order to steal the information stored in or communicated to the device. As a result, the power overhead for implementing these layers of security is approximately 13% over the basic tag controller logic.

Some theoretical traffic models, such as ALOHA traffic models [13, 14, 15],were developed to describe RFID communication patterns. However, those traffic models are not capable of providing precise views of RFID-command exchanges among interrogators and tags. In order to provide a precise power profile of RFID communication for power consumption analysis, a *Traffic-profiler Platform* has been studied in this dissertation. It is capable of capturing the RFID command traffic, analyzing RFID commands, and generating the traffic profile to a readable text file.

Due to the expanding usage of RFID technology, many RFID standards have been developed in different countries for different needs. In order to either reduce the design cost (time-to-market) for a variety of RFID applications or meet the low-power requirement, this dissertation introduces a *Physical Layer Design Automation* tool which generates encoding/decoding blocks in hardware. Based on different waveform descriptions of a variety of RFID standards, it is capable of generating corresponding hardware encoding/decoding blocks. The experimental results demonstrate that the power consumption of five encoding/decoding benchmarks are less than 5 μW. In addition, the difference of the power consumption between the encoding/decoding blocks generated by the tool and those generated by hand is within a range of 0.6 μW at most.

1.4 OUTLINE OF THE DISSERTATION

- Chapter 2 introduces reviews the prior research, Passive Active RFID Tags (PARTs), including its architecture, and a power reduction component, the burst switch. The concept of discrete-time Markov process models used for modeling the RFID systems and energy analysis is described in this chapter.
- Chapter 3 describes the first contribution, the *Smart Buffer* technique, and its architecture. The *Smart Buffer* allows a tag to remain in a sleep/standby mode until addressed. This chapter also illustrates a RFID testing circumstance in a testing scenario and an associated Markov Process

model to analyze the efficiency of power saving. Detailed energy analysis for active RFID tags with or without using *Smart Buffer* is presented and the comparison of power consumption is provided in this chapter.

- Chapter 4 shows the second contribution, multi-layer low-power security architecture, providing data protection transcending the passive-transaction, physical, and data layers to provide secure transactions. The experimental results demonstrate the implementation of our encryption algorithm has resistance to the side-channel attacks.
- Chapter 5 describes the third contribution, an FPGA-based traffic profiler system to generate traces of RFID communications for both tag verification and power analysis without the need of actual hardware. It demonstrates how to use the generated traffic trace file to profile the power consumption of the communication.
- Chapter 6 depicts the fourth contribution, a design automation technique, for generating encoding and decoding hardware blocks of the physical layer design in RFID systems. The main benefit of creating this tool is to shorten the design time for a variety of RFID standards and applications.
- This research is concluded in Chapter 7. Future research directions are proposed.

2.0 BACKGROUND AND PRIOR WORK

This chapter presents a general background relevant to this dissertation research and reviews prior work developed by the RFID research group of the RFID Center of Excellence in University of Pittsburgh. The related background to each contribution is presented in the beginning of each contribution chapter. Section 2.2 presents the prior research on the Passive Active RFID Tag (PART) architecture. This section gives background on the main components of the PART architecture, including a description of the burst switch in Section 2.2.1. Section 2.3 describes the background of a discrete-time Markov process, which is utilized to analyze the power consumption trends of the *SmartBuffer* technique.

2.1 INTRODUCTION TO RFID SYSTEMS

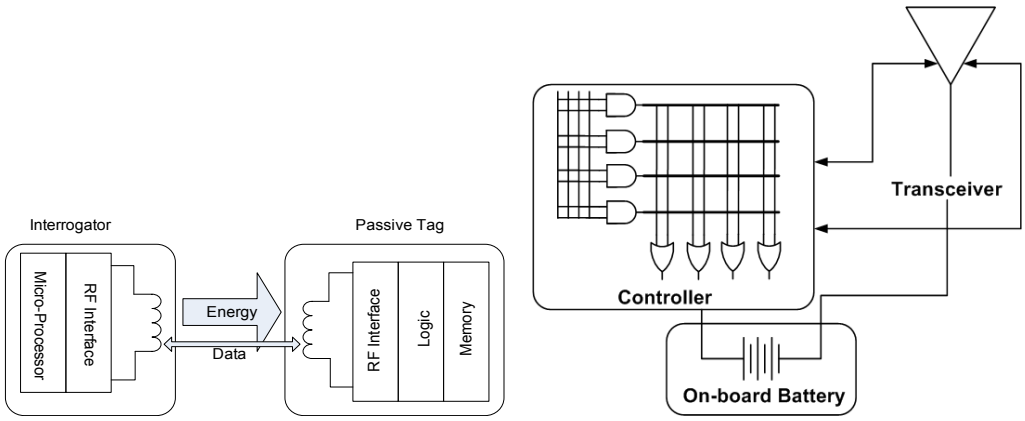
The Radio Frequency IDentification (RFID) technology is an automatic identification technology that uses radio waves carrying energy or data between interrogators or unidentified tags/objects. During World War II, the radar technology was invented to discover air crafts. In the early 1940s, the Identification Friend or Foe (IFF) system, the first RFID-based application, was used to identify allied planes by sending out radio waves and detecting the echoes [17, 18]. Since the first RFID-based access control system invented by Charles Walton in 1973 [17], the RFID technology has been studied in its applications. Due to its particular identification method, the application of RFID technology is expanding in many area. For example, tracking products, supply chain, asset management, and animal tracing [17, 1, 19, 6] are low-security level applications and access control, wireless payment,confidential cargo shipment and homeland security [20, 21] are high-security level applications.

An RFID system contains two main components: interrogators and RFID tags. Active and passive are two dominant types of RFID tags shown in Figure 5. A passive RFID tag is also known as a passive transponder. In general, a traditional passive RFID tag usually consists of a transponder antenna, RF interface, logic circuitry and memory shown in Figure 5(a) [1, 2]. It does not have an internal power supply for the logic circuitry and memory. In other words, the passive transponder can operate only when the supplied energy is transferred from an interrogator via an alternating magnetic field. Thus, the antenna and the RF interface absorb the energy and then generate an induced voltage to perform memory access. After accessing the memory, the passive tag backscatters the energy back to the interrogator and modulates the energy signal with Amplitude-Shift Key (ASK) or On-Off Key (OOK) modulation.

Many standards of passive RFID tags are developed for a variety of applications. For example, the ISO 18000-6C standard [22], issued by International Standard Organization (ISO), and the EPC Gen2 Standard [23] were developed for item management to replace the bar code. Not only can the passive tag be used for identifying or tracing ground items, but they also can be applied to aircraft components, such as Aerospace Standard AS5678, a standard passed by Society of Automotive Engineers (SAE) for air craft. Furthermore, the passive tags have also been employed to applications that require high personal privacy, such as wireless payment [24] or biometric passport [21].

Unlike a passive RFID tag, an active RFID tag has an internal power supply, usually a battery. In general, a traditional active RFID tag is composed of an RF transceiver, a processor-based controller, and an on-board battery as shown in Figure 5(b). The transceiver includes a transmitter and a receiver. The transmitter is powered to transmit responded signals back to an interrogator with a long distance (around 300 feet) and the receiver is powered on all the time to receive incoming signals from an interrogator. The processor-based controller is responsible for processing incoming RFID commands and data and generating the associated responses. The on-board battery powers the RF transceiver and the controller.

In 2004, the ISO ratified the ISO 18000 part 7 standard as an active RFID standard. It defines the physical layer protocol, data coding mechanism and RF communication at 433 MHz. Due to an active RFID tag is energized by an internal power source, it could perform complex computation, such as cryptographic computation, and long distance transmission. For example, in 2006, the



(a) Passive RFID tag. (b) Active RFID tag.

Figure 5: An architectural overview of RFID tags.

U.S. Department of Defense was pushing to use active RFID technology to track military goods and cargo containers in worldwide [25].

Although passive RFID tags are energized by an interrogator, the shortage of induced power limits the complexity of logic computation. On the other hand, an active RFID tag has an internal power source to support strong logic computation. However, powering an active transceiver all the time and energizing complex computation limit the usage life of active RFID tags.

Energy saving and strong security features are two main study topics for the next generation of RFID technology. Due to increasing privacy requirements and secure applications, such as access control, wireless payment, and homeland security [20, 6, 21, 24], current RFID tags are not sufficient to meet these requirements. Consequently, additional security features such as data encryption and safe transactions are required. However, RFID tags have extremely strict low-power consumption requirements. Additionally, even for some particular applications requiring the capability of an active RFID tag, it is difficult to perform the battery replacement. For example, the oversea shipment is a low-cost, large-scale application requiring active tags for tracing cargos, it is not feasible to replace the battery of an active tag while the container is in the middle of nowhere. Thus, to hurdle the power limitation and improve secure features are two main problems for the next generation RFID tags.

2.1.1 Current Research Survey on Hurdling Power Limitation

Current research in the active RFID area focuses on protocol, technology, and network architecture approaches to hurdle the power limitations. From the protocol perspective, Nilsson *et al.* [26, 27] describe how to design a protocol for an active RFID system to obtain the better energy efficiency. Bhanage and Zhang proposed a tag reading protocol, Relay-MAC, to achieve reliable and energy-efficient tag reads [28]. Based on their research, power conservation can be achieved by reducing the amount of information sent over the network and providing collision detection by modifying protocols. It is possible to conserve energy for active RFID tags if a protocol provides collision-free communication or collision protection mechanisms. Authors evaluate the energy savings from reducing the number of retransmissions, however, the receivers of the active RFID tags are powered on and consume energy all the time.

Kaya and Koser study a power conservation solution for RFID systems from the technology perspective. They proposed a novel technology for RFID system, called a batteryless solution which is based on a MEMS (Micro-Electro Mechanical Systems) energy scavenger [29]. The embedded MEMS accelerometer is capable of generating voltage while a sudden or continuous acceleration happens. This technology is used in a passive RFID transponder. It can generate power for a passive RFID tag for writing the memory bits without harvesting energy from the RF signal. Therefore, it behaves like a batteryless active RFID transponder. Although it is called a batteryless active RFID transponder, the technology is constructed on a passive RFID architecture.

[30] shows the benefits of energy efficiency by using the sensor-enhanced approach for active RFID systems. With embedded range-sensor, active RFID tags control the frequency of range information exchanging among tags. While short distance between two tags, the information exchange rate will be dropped in order to conserve energy. The experimental results show that the energy consumption is reduced by 56%.

Bhanage *et al.* [13] show an approach to reduce power consumption from the network architecture perspective. They proposed an asymmetric RFID system. The power conservation can be obtained by eliminating the receiver and utilizing a simple MAC-layer protocol. The active RFID tag only transmits a tag's information, such as tag ID, but is not allowed to receive any information. They sacrifice the functionality of active RFID tags to achieve power conservation.

2.2 PASSIVE ACTIVE RFID TAGS (PART)

The RFID system contains two main components: RFID readers and RFID tags. RFID tags generally come in two types, *passive* and *active*. *Passive* tags do not contain an internal power source. As a result, these tags not only receive information from a query, they also receive energy. This energy is used to power the tag in order to determine if the tag should send a response to the query or not. While passive tags are generally cheaper than active tags, they have two major disadvantages: (1) the range of passive tags is significantly lower than that of active tags and (2) the complexity of response is significantly reduced compared to active tags due to the limited energy budget.

Unlike passive tags, *active* tags require an internal power source (usually a battery) to power the transceiver for receiving queries and transmitting responses. The power supply also powers the tag's controller, which may be an application specific integrated circuit (ASIC) or an embedded microprocessor. Since active tags contain an internal power source, they are capable of providing complex responses and transferring data within a long distance. However, their main disadvantage is the limitation of the lifetime of the internal power source. Even for some particular applications requiring the capabilities of an active RFID tag, it is difficult to perform the battery replacement. For example, oversea shipment is a low-cost, large-scale application requiring active tags for tracing cargos. It is not feasible to replace the battery of an active tag while the container is in the middle of nowhere.

To solve this problem, a novel active tag architecture was built in [31] combining passive and active tag technologies to reduce the power consumption. The goal is to build a tag system with an essentially infinite battery shelf life and an active battery life essentially equal to that of the tag. The overall architecture for such a tag is given is Figure 6. This architecture is referred to as a "Passive Active Radio Frequency Identification Tag (PART)."

Recall the architecture of a passive tag and an active tag as shown in Figure 1. In Figure 1(a), a passive tag consists of a transponder antenna (passive energy receiver) and a logic controller. In Figure 1(b), an active tag is composed of an active transceiver and a complex logic controller. The passive active RFID tag system integrates a passive energy receiver, an active transceiver, and a logic controller with two power reduction mechanisms into a novel architecture in Figure 6. Two power reduction components are a passive transceiver switch and a *Smart Buffer* circuit.

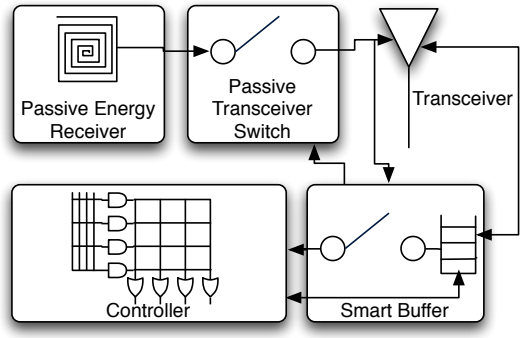


Figure 6: Overview of the ultra low-power active RFID tag.

The system contains five major components: a passive energy receiver, a passive transceiver switch, an active transceiver, a *Smart Buffer*, and a logic controller. When RF energy is received on the passive energy receiver, the passive energy receiver and the passive transceiver switch absorb RF energy to activate the active transceiver. Once the active transceiver is activated, the *Smart Buffer* begins receiving queries from an interrogator, buffering the entire incoming queries, and processing incoming commands. In the mean time, the logic controller remains in sleep mode until the *Smart Buffer* wakes it up to process a relevant RFID command. The controller may be a processor-based system or an ASIC depending on the particular application.

The operation of this architecture is as follows. When a tag is not being interrogated the battery power to the transceiver and *Smart Buffer* will be disconnected and the controller will be in the sleep mode if they are processor- or FPGA- based. On the other hand, if the *smart Buffer* or controller are ASIC or Flash FPGA- based, they will be disconnected from power. When a group of tags are to be interrogated the passive transceiver burst switch supplies power to the transceiver and *Smart Buffer*. If a particular tag is being interrogated, the *Smart Buffer* awakens and supplies power to the controller. The controller interprets the interrogation and issues the appropriate reply. The controller then returns to sleep or off mode and power is disconnected from the *Smart Buffer* and transceiver.

2.2.1 Passive Transceiver Switch

The passive transceiver switch is also called a *burst switch*. Figure 7 shows a generic circuit of a passive transceiver switch [31]. It absorbs RF energy and converts the energy into a direct current (DC). The DC behaves as a power source for waking up the rest of the tag. In the case of the PART, a battery is used to power the device, while the burst switch is used to power the battery controller. Most of the time, if there is no communication between interrogators and tags, the battery remains OFF or provides a tiny current for powering the logic controller in sleep mode. Thus, the battery life must be extended to a maximum; likely the life of the asset being tagged.

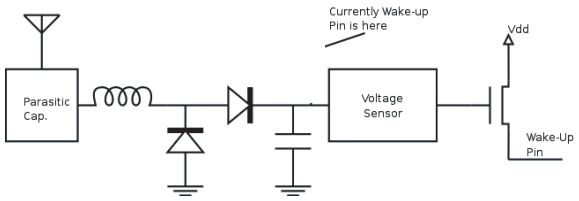


Figure 7: A simple generic burst switch.

One of the difficulties with the simple RF wake-up circuit illustrated in Figure 7 is that spurious RF energy (noise) could potentially awaken the sleeping device. Thus, it may be necessary to interface a low power or passive circuit (essentially a filter) between the RF switch and the higher power consuming receiver shown in Figure 8.

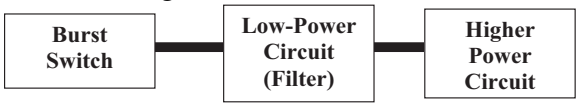


Figure 8: A low power filter to avoid false wake-up from RF noise.

The Low-Power Circuit (Filter) of Figure 8 could be any low power device that can be turned on for a short period of time, increment a counter(s) and go back to sleep. Essentially, this device acts like a receiver. A watch dog timer may be used to reset the device after extended noisy periods or after long intervals of inactivity. As an example of the operation of this low power type of device, consider the powering profile of Figure 9.

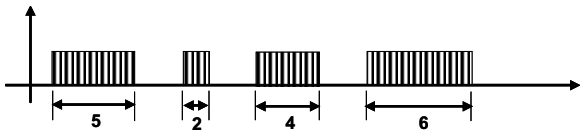


Figure 9: Powering sequence coded for security.

The code is made up of four bursts of 5, 2, 4 and 6 units of time. The device counts (possibly on a dedicated counter) the number of separate bursts; 1, 2, 3, 4; and the length of each burst; 5, 2, 4 and 6; is stored in say a register (or any suitable) memory. When the count reaches four, the registers are checked for the proper code, e.g., 5, 2, 4, 6. If the code is correct, the Higher Power Circuit is awaken. Otherwise, all of the registers are reset. This same scheme can be used for any N bursts and any length ON periods as well as any frequency or combination of frequencies or sequences. More details on the burst switch encoding are described in Section 4.3.1, **Passive Activation Layer Security**.

2.3 DISCRETE TIME MARKOV PROCESS MODEL

A discrete-time Markov process (DTMP) is a stochastic process studying how the consequence of the current status is influenced by the previous status [32]. A discrete-time Markov process is used to model the behavior of a system. In this dissertation, a discrete-time Markov process is utilized to analyze the power consumption of RFID systems. Two power consumption models are developed in Chapter 3. The first model describes the power consumption of a traditional RFID tag. The second model describes the power consumption of an RFID tag with a *Smart Buffer* .

Definition 1: A discrete-time Markov process is a stochastic process $\{X_0,\ X_1,\ X_2, ..,\ X_n, ...\}$ in discrete time $\{n = 0, 1, 2, ...\}$, where s_n denotes the *state* at discrete time *n* and a set of *states*, defined as $S = \{s_0,\ s_1, ...,\ s_n,\ s_{n+1}\}$, which is formed as [33]:

$$\mathbf{Pr}(X_{n+1} = s_{n+1}|X_n = s_n, X_{n-1} = s_{n-1}, ..., X_0 = s_0) = \mathbf{Pr}(X_{n+1} = s_{n+1}|X_n = s_n), \quad \forall n \geq 0, \forall s_{n+1}, ..., s_0 \tag{2.1}$$

Eq. (2.1) represents that moving to the next state, $X_{n+1} = s_{n+1}$, only depends on the current state, $X_n = s_n$, and is independent of any past state, $X_{n-1} = s_{n-1}, ..., X_0 = s_0$. This property is called the Markov property or Memoryless property.

Definition 2: A transition probability is the probability of moving from state s_i to state s_j at the discrete time n, which is denoted by $p_{ij}(n)$ [33]. Therefore, $p_{ij}(n) = \mathbf{Pr}(X_{n+1} = s_{n+1}|X_n = s_n)$.

If a transition probability is independent of discrete time, a discrete-time Markov process is a *homogeneous* discrete-time Markov process [33]. Therefore, the transition probability of moving from state s_i to state s_j can be denoted by p_{ij} and written as:

$$p_{ij} = \mathbf{Pr}(X_{n+1} = s_j|X_n = s_i) \tag{2.2}$$

Definition 3: A transition probability matrix P is defined as a matrix containing all p_{ij} [34]. So that:

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots \\ p_{21} & p_{22} & \cdots \\ \vdots & \vdots & p_{ij} \\ \vdots & \vdots & \end{pmatrix}, and \sum_j p_{ij} = 1, \forall i = 0, 1, 2, ... \tag{2.3}$$

A discrete time Markov process can be represented by using a state diagram composed of a set of states and a set of transitions. Figure 10 illustrates a two-state discrete-time Markov process including two states, $\{s_0, s_1\}$, and four transitions, $\{p_{00}, p_{01}, p_{10}, p_{11}\}$. For example, the probability of moving from state s_0 to state s_1 is p_{01}, and its value is α. The probability of staying within state s_0 is p_{00}. Based on Eq. (2.3), the summation of all outgoing probabilities from state s_0 is 1. Thus, $p_{00} = (1 - \alpha)$.

The transition probability matrix $P_{two-state}$ can be shown as a 2x2 array in Eq. (2.4). An "answering the phone" event is used as an example to demonstrate a discrete-time Markov process model. Bob has a mobile phone. If a phone call comes in, he will answer the phone with 90%

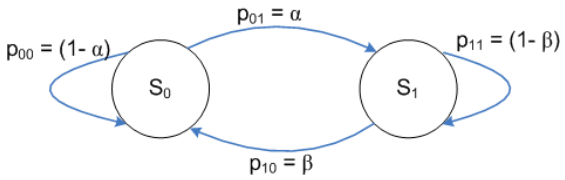


Figure 10: The state diagram of Two-state discrete-time Markov process.

probability. For all of the phone calls he answered, 20% of incoming calls are wrong-number calls. Therefore, this two-state matrix can be rewrote as $P_{two-state} = \begin{pmatrix} 0.1 & 0.9 \\ 0.2 & 0.8 \end{pmatrix}$.

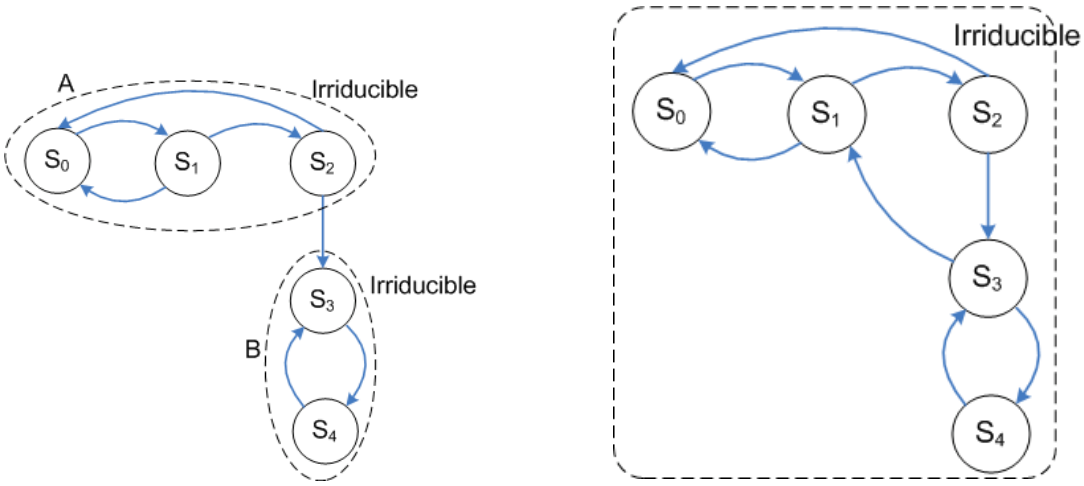
$$P_{two-state} = \begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix} = \begin{pmatrix} (1-\alpha) & \alpha \\ \beta & (1-\beta) \end{pmatrix} \tag{2.4}$$

2.3.1 Aperiodic and Irreducible Discrete Time Markov Process

Definition 4: A discrete-time Markov process is **irreducible** if all of its states communicate with any other state [34].

Figure 11 illustrates two examples of a non-irreducible and an irreducible discrete-time Markov process. The example in Figure 11(a) shows a discrete-time Markov process containing five states, $\{s_0, s_1, s_2, s_3, s_4\}$. Subset A in the graph is composed of $\{s_0, s_1, s_2\}$, where all of the states are capable of communicating with each other. Similar to Subset B in Figure 11(a), it containing $\{s_3,s_4\}$ is an irreducible set of states. However, the discrete-time Markov process containing five states in Figure 11(a) is not irreducible. State s_0 is able to reach states s_3 and s_4, but state s_0 is not accessible from states s_3 and s_4.

On the contrary, the discrete-time Markov process in Figure 11(b) is irreducible. Figure 11(b) presents a similar state diagram with additional transitions from state s_3 to state s_1. This transition makes state s_0 accessible for from states s_3 and s_4. Consequently, the entire state diagram becomes irreducible states.



(a) Subset A and B are irreducible.

(b) An irreducible discrete-time Markov process.

Figure 11: Non-irreducible and Irreducible DTMP.

Definition 5: The **distance** of state *j* is the number of **transitions** of a loop outgoing from state *j* and looping back to itself.

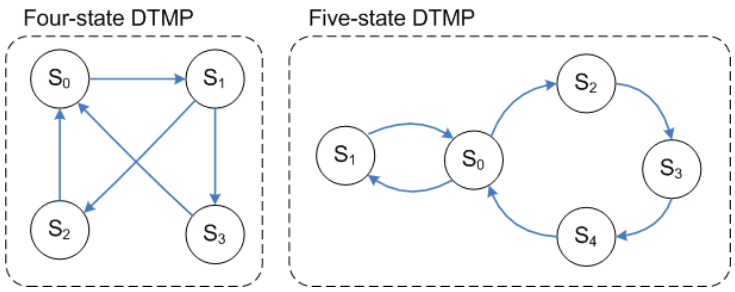
Definition 6: The **period** of state *j* is the *greatest common divisor (GCD)* of the set of distances of state *j* [34].

Definition 7: A discrete-time Markov process is **aperiodic** if it has **period** of one [34].

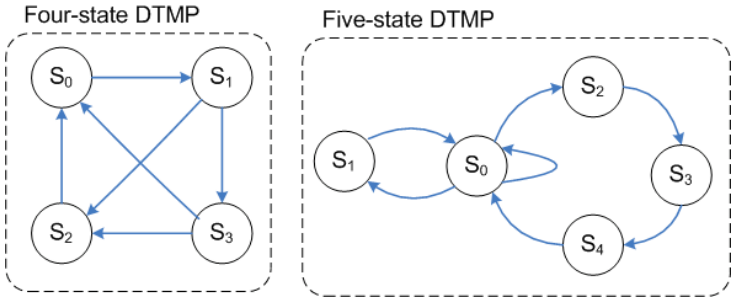
Figure 12 illustrates two examples of a periodic and an aperiodic discrete-time Markov process. The example in Figure 12(a) shows two DTMP state diagrams, four-state and five-state DTMP. For the four-state DTMP, two loops pass though state s_0. The first loop is constructed of $\{s_0,\ s_1,\ s_2\}$ and the second path consists of $\{s_0,\ s_1,\ s_3\}$. Thus, state s_0 has two *distances* $(3, 3)$. In addition, the *period* of states s_0 is three. State s_1 is similar to state s_0. The two *distances* of state s_1 are $(3, 3)$. The *period* of state s_1 is three. Therefore, the four-state DTMP is periodic.

Another example in Figure 12(a) is the five-state DTMP. The first loop of state s_0 is $\{s_0, s_1\}$, and the second loop is $\{s_0, s_2, s_3, s_4\}$. Thus, the distances of state s_0 are (2, 4). The period of the five-state DTMP is two. It is a periodic DTMP.

In Figure 12(b), the four-state and five-state DTMPs are modified by adding one more transition. The four-state DTMP has an additional transition from state s_3 to state s_2. The five-state DTMP has an additional loop on state s_0. For the four-state DTMP, this modification causes the *distances* of state s_0 to become (3,3,4). Therefore, the *period* of the four-state DTMP is one. It becomes an aperiodic discrete-time Markov process. For the five-state DTMP, the *distances* of the state s_0 become (1,2,4). Thus, the *period* of the five-state DTMP is one. It becomes an aperiodic discrete-time Markov process.



(a) Periodic four-state and five-state DTMP.



(b) Aperiodic four-state and five-state DTMP.

Figure 12: Periodic and Aperiodic DTMP.

2.3.2 Stationary Distribution

Definition 8: A stationary distribution $\Pi = \Pi \cdot P$ can be calculated by the equation shown in Eq. (2.5), if the discrete-time Markov process is an irreducible and aperiodic DTMP [33].

$$\Pi = \Pi \cdot P = \begin{bmatrix} \pi_0 & \pi_1 & \pi_2 & \pi_3 \end{bmatrix} \cdot \begin{pmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{pmatrix}, \; and \sum_{all j} \pi_j = 1 \tag{2.5}$$

A state probability represents the probability of staying in a particular state when the system is in a "long-run" time. π_0 represents a state probability of state s_0. When a discrete-time Markov process is aperiodic and irreducible, the limit of the state probability exists in a "long-run" time. A vector of state probabilities can be determined by using Eq. (2.5). It is also called a stationary distribution, Π.

2.3.3 Reward Matrix

A reward matrix is a matrix containing reward values for each transition in the state diagram of a discrete-time Markov process. This matrix assists a discrete-time Markov process in modeling a system with experimental data. This work has been demonstrated by Hawrylak in his dissertation [35]. A reward matrix represents an energy consumption matrix used in benefit analysis of using *SmartBuffer* technique in Section 3.4.

3.0 SMART BUFFERING TECHNIQUE

3.1 INTRODUCTION

Power consumption affects both passive and active devices. For an active device, the amount of power/energy required by the tag dictates the lifetime that the tag may operate. Passive devices' ranges and complexity of computation for features such as added security or access to sensors are directly impacted by power consumption. This chapter presents techniques and architectures that are applicable to active tags. These techniques are designed to address the concerns of reducing power in RFID systems without compromising their capability or to extend their capability in a power efficient manner.

To reduce the power consumption of the tag microcontroller, it is necessary to decrease the time that the controller is active. Therefore, an RF transceiver coprocessor was created to manage the buffering of messages to and from the transceiver, and the activation of the controller in order to respond to RFID primitives when necessary. This is an information filter used for power reduction. Thus the RF transceiver coprocessor, or *Smart Buffer*, assists the embedded controller to sleep or stay in a low-power idle mode while any non-relevant packets arriving at the RFID tag are ignored and valid packets wake up the processor to respond appropriately [36, 37].

The *Smart Buffer* technique allows the embedded microcontroller to sleep or stay in a low-power idle mode while any irrelevant packets arriving at the RFID tag are ignored. Meanwhile, valid packets wake up the processor to respond appropriately. Effectively, it acts as a packet filter. Before waking up the microcontroller, it analyzes every incoming command and decides whether or not to wake up the microcontroller to process the command.

A description of the *Smart Buffer* technique for a Passive Active Radio frequency Tag (PART) is given in this chapter. The smart buffering technique allows a tag to remain in a standby (sleep)

mode until addressed. For a traditional active tag, the battery life is significantly impacted by the frequency of tag awakening and the period of data processing. Thus, this smart buffering technique reduces the wake-up frequency and usage period of an active tag. The remainder of this chapter is organized as follows: Section 3.2 describes how RFID communication impacts on the power consumption of an RFID system. Section 3.3 describes the architecture of the smart buffering technique including the algorithm of smart buffering technique and main components. In Section 3.4, the Markov process model characterizes a power consumption model of *Smart Buffer* circuitry by exploiting a general scenario. In addition, it is exploited to explain how the smart buffering technique can reduce the power consumption of a PART tag. Section 3.5 reports the experimental results of power reduction.

3.2 THE IMPACT OF RFID SYSTEM COMMUNICATION

Broadcasting is the only communication method between RFID readers and tags. However, RFID technology uses not only one-to-all RFID commands but also point-to-point commands specified in ISO 18000 Part 7 and ANSI 256 active RFID standards. The ISO 18000 Part 7 standard defines 14 active RFID commands: Table 2 lists three on-to-all commands and Table 3 lists eleven point-to-point commands . The first column indicates the command code for each command, where the letters **R** and **W** represents **Read** and **Write**.

Table 2: One-to-All commands defined in ISO 18000 part 7.

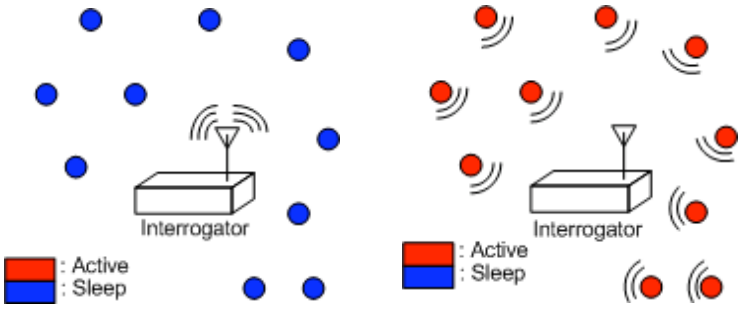
Cmd Code (R/W)	Cmd Name	Cmd Type	Cmd Description
10 / NA	Collection	Broadcast (one-to-all)	Collect all Tag IDs
11 / NA	Collection with Data	Broadcast (one-to-all)	Collect all Tag IDs with Data
14 / NA	Collection with User ID	Broadcast (one-to-all)	Collect all Tag IDs with UIDs

Three one-to-all commands, also known as **Broadcast** commands, can be found in Table 2: `Collection`, `Collection with Data`, and `Collection with UserID`. The

`Collection` command allows an interrogator to collect all Tag IDs within the interrogator RF communication range. The `Collection with Data` and `Collection with UserID` allow an interrogator to collection not only Tag IDs, but also specified data and user IDs.

Figure 13 illustrates the impacts of one-to-all communication on the power consumption for one active RFID tag. Assume that there are 10 active tags within the interrogator RF communication range. The cube represents an interrogator and circles represent tags. Initially, the microcontroller of each tag is in the **Sleep**, shown in blue in Figure 13(a).

When an interrogator broadcasts an one-to-all command, such as `Collection`, to a group of tags, all microcontrollers are aroused and go into the **Active** mode, shown in red. At this moment, the microcontroller of each tag is utilizing the peak energy to process the incoming one-to-all command and the transceiver is on for transmitting the response back to the reader as illustrated in Figure 13(b). Consequently, all tags must wake up to process the one-to-all command and respond to the reader.



(a) Before receiving a one-to-all command.

(b) After receiving a one-to-all command.

Figure 13: One-To-All communication impacts on the power consumption.

In addition to the one-to-all commands, eleven point-to-point commands are defined in the standard and listed in Table 3. They are `Sleep`, `Status`, `User ID Length`, `User ID`, `Owner ID`, `Firmware Revision`, `Model Number`, `Read/Write Memory`, `Set Password`, `Set Password Protect`, and `Unlock`. For example, if an interrogator wants to set the User ID for a particular tag, it must broadcast a `User ID` command to all tags with containing the destination tag ID. After receiving the command, the tag with the matching

destination tag ID responds to the reader and sets its User ID. But the other tags ignore the `User ID` command. Figure 14 illustrates the impact of the point-to-point communication on the power consumption. In Figure 14(a), the cube represents the interrogator and circles represents active tags. Initially, they are all in the **Sleep**, shown in blue.

Table 3: Point-to-Point commands defined in ISO 18000 part 7.

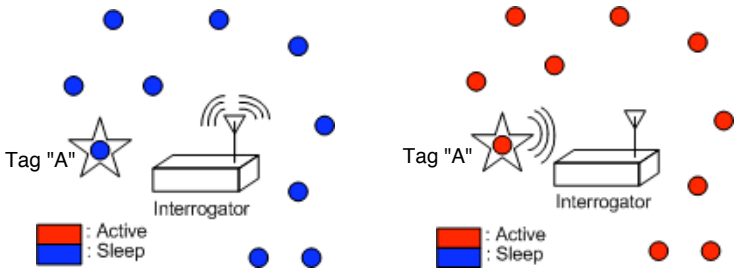
Cmd Code (R/W)	Cmd Name	Cmd Type	Cmd Description
NA / 15	Sleep	Point-to-point	Put a Tag to sleep
01 / NA	Status	Point-to-point	Retrieve Tag Status
07 / 87	User ID Length	Point-to-point	Set length of UID
13 / 93	User ID	Point-to-point	Set User ID
09 / 89	Owner ID	Point-to-point	Set Owner ID
0C / NA	Firmware Revision	Point-to-point	Set by manufacturer
0E / NA	Model Number	Point-to-point	Set by manufacturer
60 / E0	Read/Write Memory	Point-to-point	Memory data
NA / 95	Set Password	Point-to-point	Set Tag Password
17 / 97	Set Password Protect	Point-to-point	Set/Clear Tag secure bit
NA / 96	Unlock	Point-to-point	Unlock password protection

When the interrogator broadcasts a point-to-point command, such as `User ID`, to all tags, all of the microcontrollers are aroused and go into **Active** mode, shown in red. At this moment, the microcontroller is utilizing the peak energy to process the incoming point-to-point command as described in Figure 14(b). Tag A, marked with the star in Figure 14(b), is the only tag must respond the corresponding data back to the reader The rest of the tags ignore the `User ID` command and remain in **Active** mode.

The ideal communication pattern for minimal power consumption in active RFID systems requires the following behavior:

- One-to-all commands need all tags to wake up and respond.
- Point-to-point commands need only one tag to wake up and respond.

For point-to-point communication, the current tag architecture is designed contrarily. When a group of tags receive a point-to-point command, all of the tags wake up. However, only one tag, Tag A, responds. The rest of the tags wasted energy to wake up and ignore the incoming command. Therefore, the *Smart Buffer* technique, an energy saving approach, for active RFID tag is proposed to solve this problem.



(a) Before receiving a point-to-point command.

(b) After receiving a point-to-point command.

Figure 14: Point-to-point communication impacts on the power consumption.

3.3 SMART BUFFER ARCHITECTURE

The *Smart Buffer* technique is designed to achieve the ideal communication pattern for minimum power consumption in active RFID systems, particularly for the point-to-point communication. The *Smart Buffer* is a dedicated hardware which behaves like a pre-processing circuitry. In order to save power, a *Smart Buffer* is developed to sit between the transceiver and the tag microcontroller as shown in Figure 15, the overall architecture diagram of the ultra low-power active RFID tag. It allows the embedded microcontroller to remain asleep while an incoming packet is **buffered**.

The burst switch, described in Section 2.2.1, is a passive receiver that is powered by RF energy from the air, which allows the active receiver to be turned completely off. Thus, the shelf life of the battery is determined primarily by the controller's sleeping power consumption. The active receiver and *Smart Buffer* are turned on only when the burst switch is activated by the wake-up signal(s) from the interrogator. The embedded microcontroller is managed by the *Smart Buffer*.

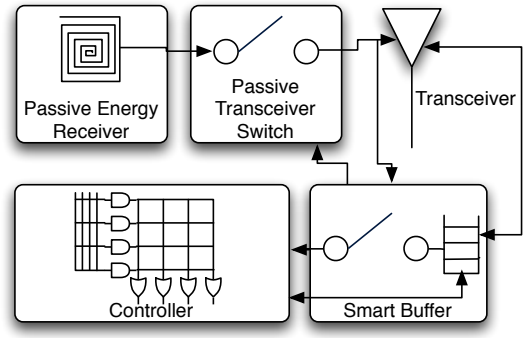


Figure 15: Overview of the ultra low-power active RFID tag.

The result is an active tag with minimal power consumption that can operate on a single battery for its lifetime.

The *smart buffer* technique allows the embedded microcontroller to sleep or stay in a low-power idle mode while any non-relevant packets arriving at the RFID tag are ignored. Meanwhile, valid packets wake up the processor to respond appropriately. Effectively, it acts as a packet filter. Before waking up the microcontroller, it analyzes every incoming command and decides whether or not to wake up the microcontroller to process the command.

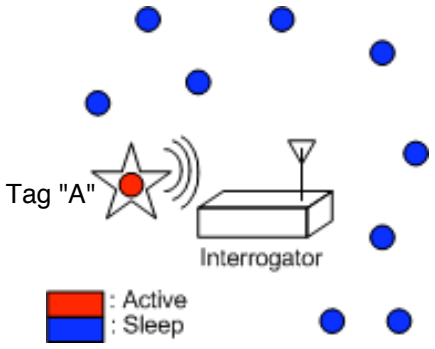


Figure 16: The impact of the *Smart Buffer* technique on the power consumption of point-to-point communication.

Figure 16 illustrates the impact of the *Smart Buffer* technique on the power consumption of point-to-point communication. The diagram shows that only one tag wakes up and dissipates the peak power for a microcontroller in Active mode. The rest of the tags remain in Sleep mode.

The *Smart Buffer* technique is capable of saving power by reducing the frequency of waking up a microcontroller. However, the drawback of using this technique is the need for an extra circuitry and power to support it. This chapter discusses the tradeoff and the efficiency of power saving of utilizing the *Smart Buffer* technique.

The top-level diagram of the *Smart Buffer* is depicted in Figure 17. The *Smart Buffer* has four I/O pins to the RF front end circuit.

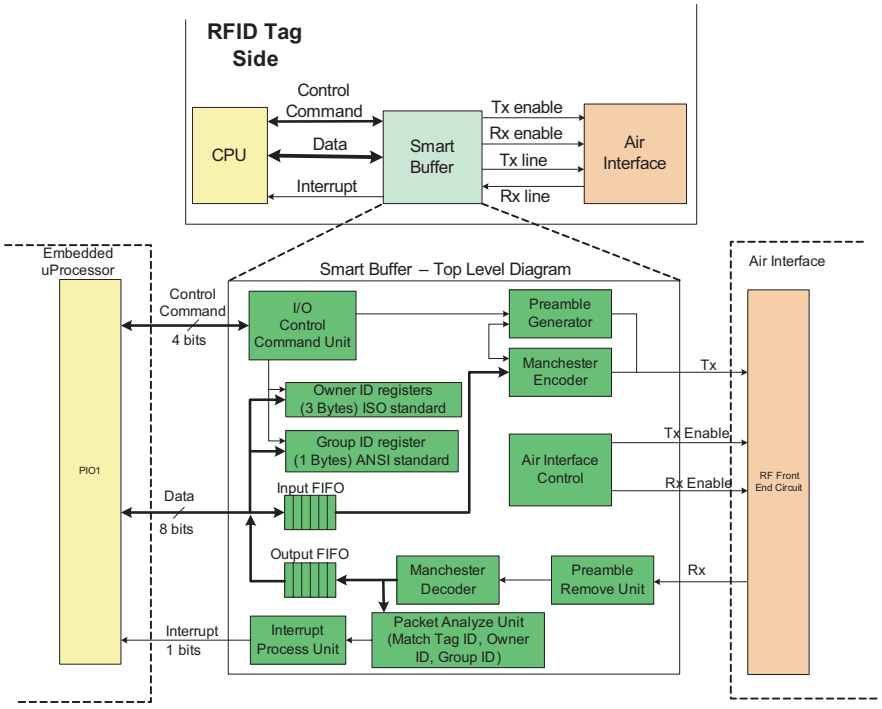


Figure 17: The top-level block diagram for the *Smart Buffer* architecture.

The blocks described in Figure 17 are enumerated as follows:

Preamble Removal Unit: Detects the incoming preamble signal and differentiates between signals intended for tags and readers.

Manchester Decoder: Converts Manchester code into binary values.

Preamble Generator: Generates the Manchester code for the tag response.

Packet Analysis Unit: Detects ID flags to determine whether to wake up the processor.

Interrupt Process Unit: Generates an interrupt to the processor.

Processor Control Command Unit: Communicates data to and from the processor.

Air Interface Unit: Communicates data to and from the air interface.

When an incoming command arrives at the RF front end circuitry, the preamble removal unit detects a valid incoming preamble signal. After the whole command is **buffered**/stored in the *Smart Buffer*, the Manchester decoder translates the Manchester encoded data immediately following a valid preamble. Based on the header information in the RFID packet, the packet analysis unit determines specific characteristics of the packet needed to decide whether to wake up the controller for response generation. This is the key procedure. If the header of the packet carrying wrong information, such as non-relevant Tag ID or User ID, the *Smart Buffer* will drop the packet.

The interrupt process unit wakes up the controller only when a complete RFID packet is stored into the output FIFO and the analysis result forwarded by the packet analysis unit is positive. The command control unit fetches control commands from the controller, such as read and write data to FIFO, update Tag ID or Group/Owner ID, and so forth. The preamble generator generates a preamble signal followed by the final sync pulse. When the final sync pulse is transmitted, the preamble generator informs the Manchester encoder to begin to output its serial encoded data. The air interface unit has output control signals which are enabled when the *Smart Buffer* needs to listen for incoming signals from the air, or when it is ready to transmit a response to readers [37].

3.3.1 Algorithm for Smart Buffer Technique

The conceptual flow in Figure 18 shows the mechanism for the RF transceiver co-processor, i.e., the *smart buffer*. In the first four states highlighted as green, the *Smart Buffer* verifies the message preamble and buffers the incoming packet. The smart buffer then checks to see if the packet was intended for this particular tag. The processor is not used to make the check, it is done in hardware while the processor remains idle or asleep. As a result, only the *Smart Buffer* consumes power. If the incoming packet is invalid, the *Smart Buffer* will ignore it and go back to the sleep state.

If the incoming packet is identified as an intended packet for the Tag, by matching the tag ID or group ID from the packet, the *smart buffer* will wake up the processor to process the packet. The processor reads data from the *Smart Buffer* and responds correspondingly.

After the processor writes response data back to the *Smart Buffer*, the *Smart Buffer* generates a bit stream of data consisting of a preamble signal and the response data with Manchester coding. It is at this point that the processor returns to the low-power mode. Upon completion of sending the packet response, the *Smart Buffer* returns to listening for the next preamble.

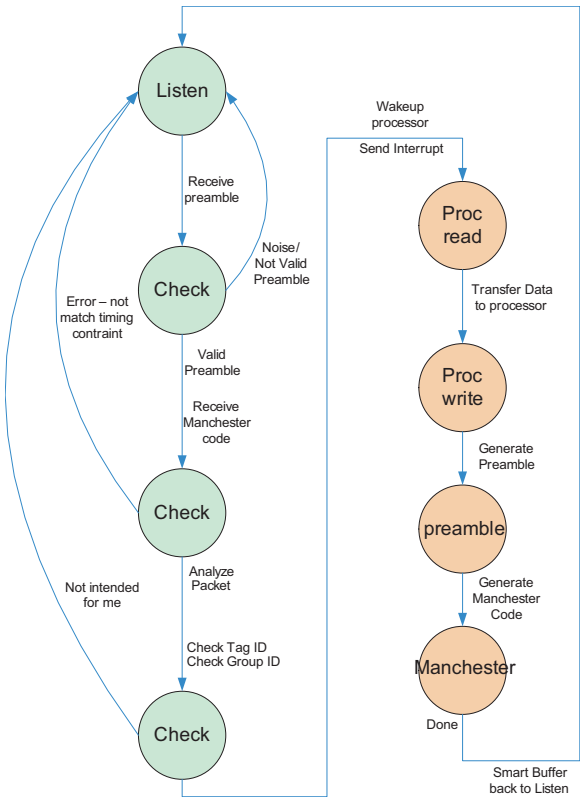


Figure 18: The conceptual flow of the *Smart Buffer*.

3.3.2 Preamble Removal Unit

The preamble removal unit detects a valid incoming preamble signal. It receives bit stream data from the RF front end circuit. This digital bit stream data is converted from an analog signal the RF circuitry receives from the antenna. However, because the input signal is analog, a signal may be due to noise rather than a preamble. Therefore, the *Smart Buffer* has the ability to tolerate noise and only recognize the valid preamble signal. The specification of the preamble signal is defined in the ANSI and ISO standards [4, 3].

The preamble signal begins with a series of pulses with 30 μs high followed by 30 μs low. Every preamble signal, regardless of whether it has originated from a tag or a reader has 20 of these regular pulses. These 20 regular pulses are followed by the final sync pulse, which determines whether a tag or reader originated the preamble. If an RFID packet comes from a tag, the final sync pulse is 42 μs high and 54 μs low. If an RFID packet comes from a reader, the final sync pulse is 54 μs high and 54 μs low. The *Smart Buffer* can ignore RFID packets which come from other tags by checking the length of the final sync pulse. It will only focus on RFID packets from interrogators as long as the sync pulse is 54/54 pattern. Other packets are not buffered.

The implementation of the preamble removal unit uses four times oversampling within each 30 μs period for each pulse. The design utilizes counters to count the sampling period. The difference in the sync pulse can be detected by counting for how many samples the final signal is high, 5 for tags and 7 for readers.

3.3.3 Manchester Decoder

The Manchester decoder translates the Manchester encoded data immediately following a valid preamble. It is the block for filtering and buffering incoming RFID packets. The decoder extracts a bit stream of Non-Return to Zero (NRZ) data from the encoded data.

The Manchester code combines the concept of clock with synchronous data into a single serial data stream as shown Figure 19. In order to enforce synchronization, Manchester code contains a transition in the middle of each Manchester bit. The Manchester bit represents zero (0) NRZ data if this transition is from high to low. Similarly, the Manchester bit represents one (1) NRZ data if the transition is from low to high. By representing data with a guaranteed transition for each bit, slight discrepancies of timing can be tolerated without disrupting the communicated data. The timing specification of the Manchester code is defined in the ISO and ANSI standards.

The Manchester decoder block converts eight serial decoded data bits into a parallel 8-bit datum (e.g. byte). In addition, the Manchester code contains a ninth bit for synchronization which is always '0'. This is called a stop-bit, and is removed during decoding. Each byte is stored in an output FIFO shown in Figure 17. When the decoder detects the final bit of an RFID packet, it stops storing data into the FIFO and asserts the end of packet (EOP) signal to the interrupt process unit.

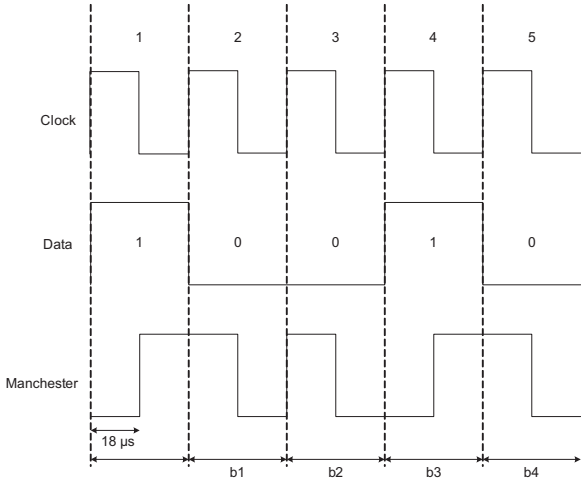


Figure 19: Example of Manchester encoding.

Based on the analysis result, the interrupt process unit determines if it is necessary to wake up the processor.

3.3.4 Packet Analysis Unit

Based on the header information obtained from the RFID packet, the packet analysis unit attempts to determine the specific characteristics of this particular packet needed to decide whether to wake up the processor for response generation. These elements include (1) the primitive operation code (or opcode), (2) the tag id for which the packet is intended, and/or (3) some other distinguishing information or id from the packet.

Both the ISO and ANSI standards have their own algorithms for tags to identify RFID packets. Therefore, the packet analysis unit has the ability to switch between these two algorithms seamlessly. For example, if the tag travels to Europe or Asia where the RFID system follows the ISO standard, the *Smart Buffer* can perceive that it being accessed by ISO primitives. However, if the tag returns to the United States, it would switch over to recognizing ANSI commands.

For broadcast commands, both standards allow partitioning of the tags into different *bins* with a unique identifier where a logical bin can contain an unlimited number of tags. This is accomplished by assigning that unique id to each tag contained within the bin. In ISO this is called the *Owner ID* and for ANSI the *Group ID*. For point to point commands, a *Tag ID* is used distinguish the destination tag.

The packet analysis algorithms are summarized in Tables 4 and 5. In both standards, commands are segregated into broadcast commands (B) and point to point (P2P) commands. If the tag receives an ISO broadcast command, it always responds if no Owner ID is set within the tag. If an Owner ID is set, the tag only responds if an Owner ID is included in the command and it matches the stored Owner ID. For ANSI, broadcast commands are subdivided between 3 opcodes, 30, 16, and 35. If the opcode is either 16 or 35, the tag always responds. If the opcode is 30, the tag only responds if the internal Group ID matches the primitive Group ID. ANSI specifies that a stored Group ID of zero (0) always results in a match. For point to point commands, both standards require a Tag ID match. However, ISO requires that the Owner ID must match as well as the Tag ID for point to point commands with an Owner ID present in both the tag and command.

Once the packet analysis unit verifies that the packet requires a response, it sends a signal to the interrupt process unit to process the packet stored in the FIFO. Because the packet analysis occurs in parallel with the packet buffering, a signal can be sent to the interrupt unit prior to the entire packet being buffered. If the packet does not require processing it is dropped from the FIFO. This prevents the processor from being powered up unless it is needed to process the packet.

3.3.5 Interrupt Process Unit

The interrupt process unit will wake up the processor only when a whole RFID packet has been stored into the Output FIFO and the analysis result forwarded by the packet analysis unit is positive. Figure 20 shows the state diagram for the interrupt procedure. The interrupt process unit will go back to the idle state after sending an interrupt signal to the processor.

Table 4: Packet analysis algorithm for the ISO standard.

Input			Output
B/P2P	Owner ID field	Owner ID in tag	Process command
B	No	No	Yes
B	No	Yes	No
B	Yes	No	Yes
B	Yes	Yes	If Owner ID match
P2P	No	No	If Tag ID match
P2P	No	Yes	If Tag ID match
P2P	Yes	No	If Tag ID match
P2P	Yes	Yes	If Tag ID and Owner ID match

3.3.6 Command Control Unit

The command control unit fetches control commands from the processor, such as read and write data to FIFO, update Tag ID or Group/Owner ID, starts the transmission procedure, and so forth. For processor compatibility, it was desirable to minimize the number of lines between the processor and the *Smart Buffer*. Thus, four (4) parallel lines are utilized to communicate the processor to buffer command control.

The processor to *Smart Buffer* commands are illustrated in Figure 21. After waking up the processor, the smart buffer listens for the processor to initiate commands for data communication. As shown in Figure 21, the control unit decodes 4-bit processor commands into 5 basic operations: transmit, update, push, pull, and null. The double circle represents a potentially multi-cycle operation.

Based on different control commands, the unit will determine the direction of the bi-directional *Smart Buffer*, processor interface. In Figure 21, `dir = 0` represents that the direction of I/O is from processor to *Smart Buffer* and `dir = 1` is the reverse. Because the *Smart Buffer* and processor are operated in two different clock domains, a hand-shaking communication approach is

Table 5: Packet analysis algorithm for the ANSI standard.

Input	**Output**
Opcode	Process command
30	If Group ID match[1]
16	Yes
35	Yes
P2P	If Tag ID match

required to push/pull data to/from the FIFOs. Therefore, it is necessary to dedicate more than one cycle to transmit a single byte of data between the processor and FIFO.

Once the processor has generated a response and completed pushing the response data into the FIFO, it sends the `transmit` command. This signals the *Smart Buffer* to generate the preamble signal, convert data in the FIFO to a serial data stream and encode the bit stream data in Manchester coding.

3.3.7 Preamble Generator

According to the ISO and ANSI standards, the preamble generator generates a preamble signal with 20 pulses of 60 μs, 30 μs high and 30 μs low, followed by the final sync pulse. Because of this RFID packet generated from a tag, the final sync pulse is 42 μs high and 54 μs low. The preamble generator utilizes several counters to trace the period time for each pulse. These counters are running with the *Smart Buffer* system clock at 33MHz. Therefore, the preamble signal will be skewed less than 1 μs requiring the reader to tolerate an error of +/- 3.3%.

In order to help the receiver filter out an ambient noise in the air, a mark state, or a stable logic low signal for 120 μs is generated and transmitted just prior to the preamble signal. While the preamble generator creates the mark state and preamble signal, the air interface unit forces the `rx_enable` signal low and raises the `tx_enable` signal.

[1] A group ID match always occurs if the currently stored group ID within the tag is zero (0).

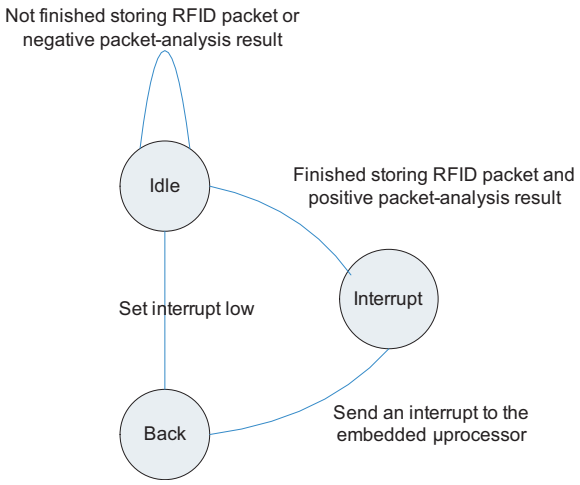


Figure 20: The interrupt finite state machine.

When the final sync pulse is transmitted, the preamble generator informs the Manchester encoder to begin to output its serial encoded data immediately. Any significant gap between data and the preamble signal may cause an error, which may not be tolerated in the system.

3.3.8 Manchester Encoder

The Manchester encoder starts encoding the data in the input FIFO after it is notified by the preamble generator. This notification occurs during the transmission of the final sync pulse to give the encoder enough time to have the first byte of data ready. First, the Manchester encoder converts the next available byte in the FIFO to eight (8) single serial bits. Those eight (8) bits are individually stored in single-bit shift registers. In addition, a stop bit needs to be appended for each byte of data in the shift registers.

The Manchester encoded data is the output of a NRZ data XOR coding clock, shown in Figure 19. The NRZ data is synchronized with the coding clock. According to the specification of ISO and ANSI standard, the period of coding clock is 36 μs; 18 μs high and 18 μs low. This clock signal is generated in the Manchester encoder block. The Manchester encoder combines the coding clock and serial data as shown in Figure 22.

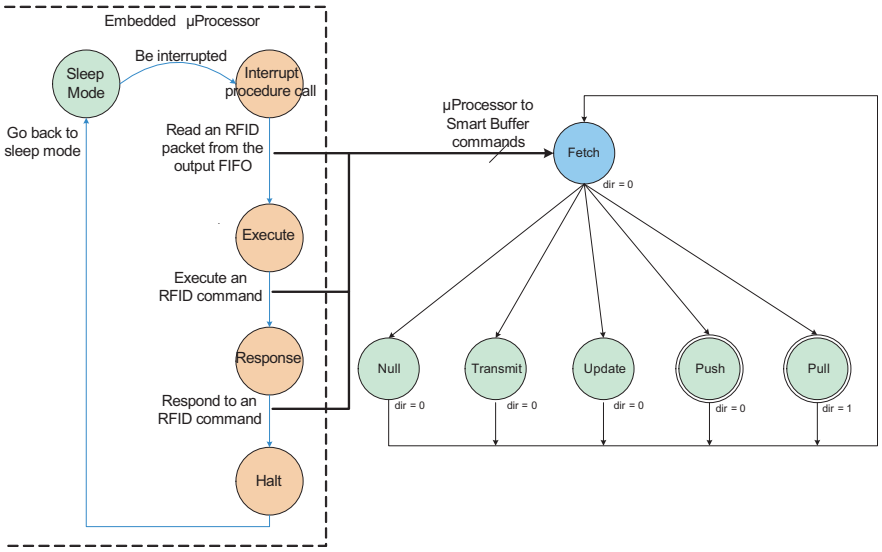


Figure 21: Fetch and operation diagram for the command control unit.

3.3.9 Air Interface Unit

The air interface unit has two output control signals, `tx_enable` and `rx_enable`. Since the *Smart Buffer* defaults to listening for incoming signals from the air, the `rx_enable` signal is set high and `tx_enable` signal is set low at all times except when the *Smart Buffer* is ready to transmit an RFID response packet back to readers.

3.3.10 Experimental Results

The *Smart Buffer* was prototyped on the Spartan 3 FPGA as well as studied for ASIC implementation using 0.16 μm OKI standard cells. While it was possible to implement FIFO blocks on the Spartan 3 FPGAs using Xilinx IP blocks, ASIC versions of these FIFOs were not available. In order to study the impact of having the *Smart Buffer* in an ASIC versus the Spartan 3 FPGA, three components were separated from the larger design and synthesized and power profiled independently. These blocks include the preamble detection, a 30 kHz wake-up signal detection, and the Manchester decoder.

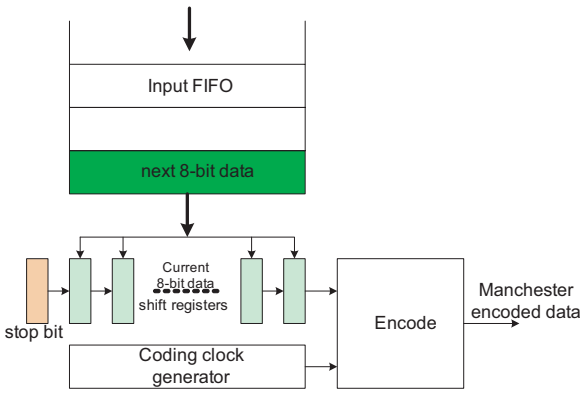


Figure 22: Top-level diagram for the Manchester encoder.

The results for power estimation of the ASIC and Spartan 3 based smart buffer components are displayed in Table 6. Both power analysis are based on post synthesis simulation with the exact same stimuli. The FPGA power results were computed using Xilinx Xpower and the ASIC power results were calculated using Synopsys PrimePower. Based on the results from Table 6 the ASIC version of the implementation uses orders of magnitude less dynamic power for all three components. Interestingly, the quiescent power alone is nearly three orders magnitude greater than the dynamic power of the ASIC.

Table 6: ASIC versus FPGA power consumption for portions of the *Smart Buffer*.

Component	**Spartan 3**	**0.16 μm ASIC**
Wake-up Signal	0.01 mW	0.001 mW
Preamble Detection	0.87 mW	0.005 mW
Manchester Decoder	0.95 mW	0.002 mW
Quiescent Power	92 mW	0 mW
Total	93.83 mW	0.008 mW

In order to operate the *Smart Buffer*, an ultra low-power active RFID tag needs to consume extra power by 8μW, which is close to the power consumption of a microcontroller remaining in the Sleep mode (1 to 15 μ A) as shown in Table 7. Table 7 lists the power consumption for

microcontrollers and the *Smart Buffer*. An active tag consumes the extra power (i.e. 0.008 mW)to avoid waking up the microcontroller frequently. According to Table 7, every time to wake up the microcontroller consumes 2 to 8 mA.

Table 7: The comparison of the power consumption among microcontrollers and the *Smart Buffer*.

Model	PIC18F6720	MSP430F16x	ATmega128L	*Smart Buffer*
Frequency (MHz)	20	8	8	10
Word size (bit)	8	16	8	8
Power (awake; mA)	2.2mA	2mA	8mA	0.008mW
Power (Sleep; μA)	1μA	1.1μA	15μA	8μW

The experimental results of the *Smart Buffer* circuitry is power consumption parameters used for the power analysis in the next section.

3.4 MARKOV PROCESS ENERGY MODEL

In order to analyze the power dissipation of each tag in detail, two different Markov-process models of a RFID transaction were built. The first model, shown in Figure 23, is for a tag with μprocessor only, which is abbreviated as the MP-μP model. The second model, shown in Figure 24, is for a tag with a μprocessor and a smart buffer, which is abbreviated as the MP-μP-SB model. Five states are used in both Markov process models: S_1, sleep; S_2, listen for preamble; S_3, receive command; S_4, process command; and S_5, transmit response. By traversing those five states, both models represent the control flow of an RFID tag for a communication transaction.

During a scenario when a tag is not participating in a transaction, the tag remains in the *sleep* state, S_1. After receiving a wake-up tone, it proceeds to state S_2 and listens for the preamble signal. The transition from S_2 to S_3 occurs when a tag receives a valid preamble and is ready to receive an incoming command. After receiving a command, the tag decides whether to continue processing the command (e.g. move to state S_4) or return to the "listen for the preamble state" (e.g. move to state S_2). If the incoming command is intended for the tag, it will process the command in state

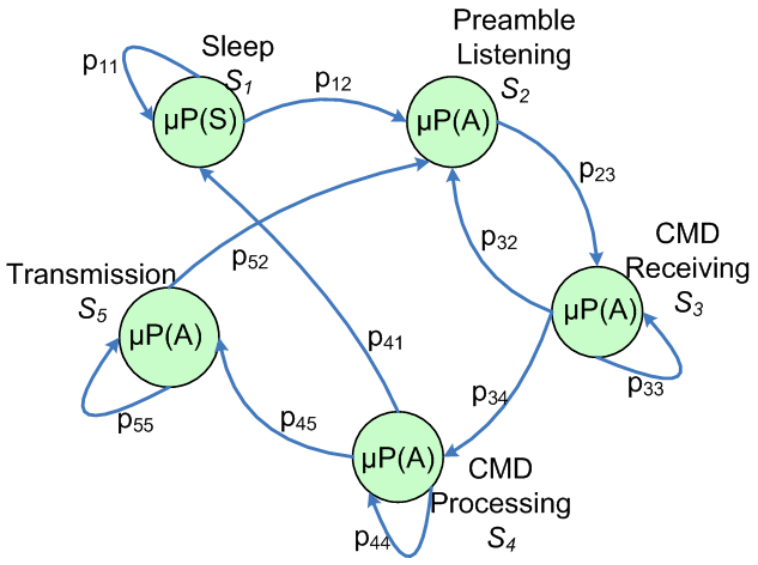


Figure 23: Markov-process model for a RFID tag with an on-tag μprocessor (MP-μP model).

S_4 and transmit the associated information back to the reader in state S_5. Otherwise, the tag will drop the command and transition from S_3 to S_2. If the following command received puts the tag to sleep, it transitions to the sleep state, S_1. The only difference between the transitions of the two scenarios is that in the MP-μP scenario the transition to S_1 occurs in S_3 and for MP-μP-SB this transition occurs in S_4. This is because the *Smart Buffer* does not currently recognize the sleep command.

The labels *μP(A)* and *μP(S)* are used to represent the μprocessor being in either the *Active* or *Standby* mode, respectively. Similarly the labels, *SB(A)* and *SB(S)*, in the MP-μP-SB model represent that the *Smart Buffer* is in either the *Active* or *Standby* mode, respectively.

In the MP-μP model, the μprocessor is in standby in the sleep state, and active in all other states. For many scenarios the on-tag μprocessor will dissipate energy when receiving incoming commands that are not intended for the tag and are eventually dropped. For example, a RFID reader sends *N* point-to-point commands equally distributed to *N* RFID tags in a reachable area. For each tag, the MP-μP model in Figure 23 indicates that a μprocessor will stay in a loop between states S_3 and S_2 for *N-1* communications. Consequently, the μprocessor dissipates energy processing *N-1*

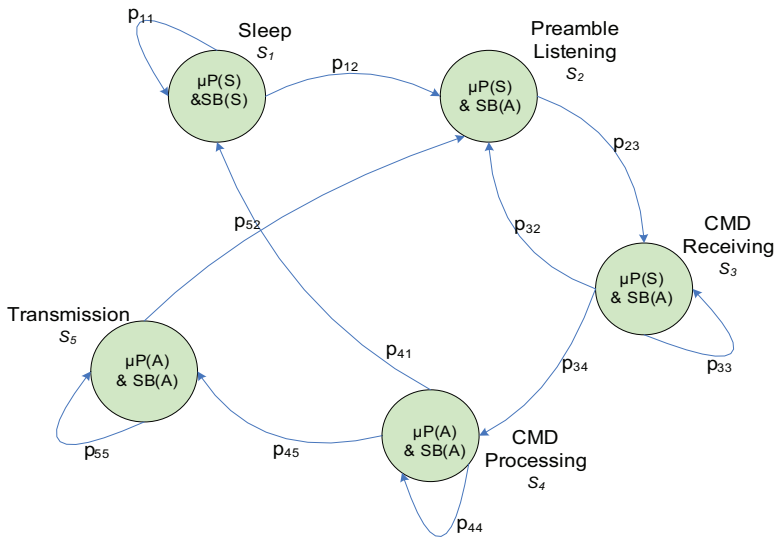


Figure 24: Markov-process model for a RFID tag with an μprocessor and a *Smart Buffer* (MP-μP-SB model).

extraneous communications.

In contrast, in the MP-μP-SB model the *Smart Buffer* alone becomes *Active* (*SB(A)*) while listening for a preamble signal and receiving a command (states S_2 and S_3), while the μprocessor still remains in *Standby* mode ($\mu P(S)$). Thus, if the incoming command is not destined for this tag, the μprocessor will never become active. This can result in a power reduction of several orders of magnitude, see Section 3.3.10.

3.4.1 Discrete Time Markov Energy Model for MP-μP Model

Figure 23 illustrates the state diagram of the discrete time Markov process μP model. The transistion matrix of the MP-μP model can be obtained in a square array as $P_{\mu p}$ in Eq. (3.1). In the transition matrix, the probabilities p_{ij} represents transition probabilities between *State i* and *State j*. If $p_{ij} = 0$, it means there is no probability of a transition between *State i* and *State j*.

$$P_{\mu p} = \begin{pmatrix} p_{11} & p_{12} & 0 & 0 & 0 \\ 0 & 0 & p_{23} & 0 & 0 \\ 0 & p_{32} & p_{33} & p_{34} & 0 \\ p_{41} & 0 & 0 & p_{44} & p_{45} \\ 0 & P_{52} & 0 & 0 & p_{55} \end{pmatrix} \tag{3.1}$$

All of the transition probabilities are determined by the following environmental parameters: the number of tags (N_{tag})within a reachable area, the number of repeat runs (N_r), the number of one-to-all (broadcast) commands (N_B), the number of point-to-point commands (N_{p2p}), and the command time (T_c). All probabilities in the matrix $P_{\mu p}$ are enumerated as follows:

- p_{11} represents the probability that the tag remains in the sleep state. It can be obtained by $1 - p_{12}$.
- p_{12} represents the probability that the tag receives an incoming preamble. The probability, p_{12}, can be obtained as $p_{12} = (T_c \cdot N_r)/T_{total}$, where T_c is composed of the preamble time, the point-to-point command time, and the one-to-all command time . The command time can be obtained by

$$T_c = T_{preamble} + N_B \cdot \tau_B + N_{p2p} \cdot \tau_{p2p}$$

- p_{23} represents the probability that the tag receives an incoming command, which always equals to one.
- p_{32} represents the probability that the incoming command is a not relevant. The tag drops the command. The probability can be obtained as $1 - (p_{33} + p_{34})$
- p_{33} represents the probability that the tag is receiving the incoming command, which can be calculated by:

$$p_{33} = \frac{\sum \Lambda_{all_incoming_C MDs} - 1}{\sum \Lambda_{all_incoming_C MDs}}$$

The $\Lambda_{all_incoming_C MDs}$ variable corresponds to the length of all commands or the processing time of all commands.

- p_{34} represents the probability that the incoming command is a relevant command. The tag attempt to process the command and generate feedback data. The probability can be calculated by $p_{34} = (1 - p_{33}) \cdot (p_B + p_{p2p} \cdot p_{relevant})$, where p_B indicates the probability of processing one-to-all commands , p_{p2p} represents the probability of processing point-to-point commands, and $p_{relevant}$ represents the probability of processing relevant commands.
- p_{41} represents the probability that the incoming command is a sleep command. Thus the tag goes into the sleep mode. The probability can be calculated by $p_{41} = (1-p_{44}) \cdot (p_{sleep} \cdot p_{relevant})$, where p_{sleep} represents the probability of processing sleep commands, and $p_{relevant}$ indicates the probability of processing relevant commands.
- p_{44} represents the probability that a command is still in process. The tag is processing the command and generating feedback data. The probability can be calculated by:

$$p_{44} = \frac{\sum \Lambda_{response_C MDs} - 1}{\sum \Lambda_{response_C MDs}}$$

  The $\Lambda_{response_C MDs}$ variable corresponds to the length of response commands or the processing time of feedback data.
- p_{45} represents the probability that the tag finished processing the command and attempts to transmit the feedback data, which can be obtained as $1 - (p_{44} + p_{41})$
- p_{52} represents the probability that the tag finished transmission and listens for the next preamble signal, which can be obtained as $1 - p_{55}$.
- p_{55} represents the probability that the tag is still transmitting data. This probability is equal to the probability, p_{44}.

For instance, the simulation scenario of this Markov power model assumes that one RFID interrogator exists and there are 20 tags within a reachable area ($N_t = 20$). As an example, the number of tags may represent the number of point-to-point commands is 20 ($N_{p2p} = 20$) if the reader equally distributes the point-to-point commands. The reader is assumed (1) to send a one-to-all command for collecting all tag IDs ($N_B = 1$), (2) to set the User ID for each tag ($N_{p2p} = 20$), and (3) to put each tag back to sleep ($N_{p2p} + 20$). The number of tags' response equals to $N_{resp} = N_{p2p} = 20$. Therefore, the total number of commands sent by the reader is $N_B + N_{p2p} + N_{sleep} = 1 + 20 + 20$ and the total commands responded from tags is $N_{sleep} = 20$.

The whole communication pattern repeats 20 times within one hour and the distribution of all commands for tags is an even distribution. After applying these environmental parameters, the transition matrix can be calculated as

$$P_{\mu p} = \begin{pmatrix} 0.774011 & 0.225989 & 0 & 0 & 0 \\ 0 & 0 & 1.000000 & 0 & 0 \\ 0 & 0.007858 & 0.992102 & 0.000039 & 0 \\ 0.000039 & 0 & 0 & 0.992024 & 0.007937 \\ 0 & 0.007937 & 0 & 0 & 0.992063 \end{pmatrix} \tag{3.2}$$

Because this matrix is an aperiodic and irreducible transition matrix. These two properties can be found in Section 2.3.1. The stationary distribution, $\Pi_{\mu p} = \begin{bmatrix} \pi_1 \ \pi_2 \ \pi_3 \ \pi_4 \ \pi_5 \end{bmatrix}$, can be calculated as:

$$\begin{aligned} \Pi_{\mu p} &= \Pi_{\mu p} \cdot P_{\mu p} \\ &= \begin{bmatrix} \pi_1 \ \pi_2 \ \pi_3 \ \pi_4 \ \pi_5 \end{bmatrix} \begin{pmatrix} p_{11} & p_{12} & 0 & 0 & 0 \\ 0 & 0 & p_{23} & 0 & 0 \\ 0 & p_{32} & p_{33} & p_{34} & 0 \\ p_{41} & 0 & 0 & p_{44} & p_{45} \\ 0 & P_{52} & 0 & 0 & p_{55} \end{pmatrix} \end{aligned} \tag{3.3}$$

Therefore, the elements of $\Pi_{\mu p}$ can be calculated as:

$$\begin{cases} \pi_1 &= p_{11}\pi_1 + p_{41}\pi_4 \\ \pi_2 &= p_{12}\pi_1 + p_{32}\pi_3 + p_{52}\pi_5 \\ \pi_3 &= \pi_2 + p_{33}\pi_3 \\ \pi_4 &= p_{34}\pi_3 + p_{44}\pi_4 \\ \pi_5 &= p_{45}\pi_4 + p_{55}\pi_5 \\ 1 &= \pi_1 + \pi_2 + \pi_3 + \pi_4 + \pi_5 \end{cases} \tag{3.4}$$

An energy matrix is a *reward matrix* introduced in Section 2.3. It is used to describe the energy characteristics of each transition for the MP-μP model in Figure 23. The energy matrix represented by $E_{\mu p}$ is given in Eq. (3.5). Each element in the matrix represents the energy consumption for each

corresponding transition. For example, the energy consumed while supporting a tag remaining in sleep mode is stored in the element ϵ_{11}. Thus, it can be obtained by $(PW_{\mu p(s)} + PW_{RF-sleep}) \cdot (T_{hour} - T_t \cdot N_r)$, where $PW_{\mu p(s)}$ indicates the power consumption of a microprocessor remaining in sleep mode, and $PW_{RF-sleep}$ represents the energy consumed when a RF front-end circuit remains in sleep mode.

$$E_{\mu p} = \begin{pmatrix} \epsilon_{11} & \epsilon_{12} & 0 & 0 & 0 \\ 0 & 0 & \epsilon_{23} & 0 & 0 \\ 0 & \epsilon_{32} & \epsilon_{33} & \epsilon_{34} & 0 \\ \epsilon_{41} & 0 & 0 & \epsilon_{44} & \epsilon_{45} \\ 0 & \epsilon_{52} & 0 & 0 & \epsilon_{55} \end{pmatrix} \tag{3.5}$$

All of the energy elements in the matrix $E_{\mu p}$ are enumerated as follows:

- ϵ_{11} represents the energy consumed by a tag remaining in the sleep state. It can be obtained by $(PW_{\mu p(s)} + PW_{RF-sleep}) \cdot (T_{hour} - T_t \cdot N_r)$, where $PW_{\mu p(s)}$ represents the power consumption of a microprocessor remaining in sleep mode, and $PW_{RF-sleep}$ indicates the energy consumed when a RF front-end circuit remains in sleep mode.
- ϵ_{12} represents the energy consumed by a tag receiving a wake-up signal. It can be calculated by $(PW_{\mu p(s)} + PW_{RF-active}) \cdot T_{wakeup} \cdot N_r$, where $PW_{RF-active}$ indicates the power dissipated when a RF front-end circuit remains in active mode.
- ϵ_{23} represents the energy consumed by a tag receiving a preamble signal. It can be calculated by $(PW_{\mu p(a)} + PW_{RF-active}) \cdot T_{preamble} \cdot N_r$
- ϵ_{32} represents the the energy consumption for a tag to realize the incoming command is non-relevant, which can be attained by $(PW_{\mu p(a)} + PW_{RF-active}) \cdot 36\mu s \cdot N_r$.
- ϵ_{33} represents the energy consumed by a tag receiving an incoming command, which can be obtained by $(PW_{\mu p(a)} + PW_{RF-active}) \cdot [(N_B \cdot (L_B - 1) + N_{p2p} \cdot (L_{p2p} - 1)) \cdot 36\mu s] \cdot N_r$, where L_B and L_{p2p} are the length of a one-to-all command and the length of a point-to-point command respectively.
- ϵ_{34} represents the energy consumed by a tag entering the processing-command state, which is equal to ϵ_{32}.
- ϵ_{41} represents the energy consumed by a tag to find out that the incoming command is a sleep command and put the microcontroller into sleep mode, which is equal to ϵ_{32}.

- ϵ_{44} represents the energy consumed by a tag generating feedback data. It can be calculated by $(PW_{\mu p(a)} + PW_{RF-active}) \cdot [(N_{B-resp} \cdot (L_{B-resp} - 1) + N_{p2p-resp} \cdot (L_{p2p-resp} - 1)) \cdot 36\mu s] \cdot N_r$, where L_{B-resp} and $L_{p2p-resp}$ are the lengths of the response data for a one-to-all command and a point-to-point command respectively.
- ϵ_{45} represents the energy consumed by a tag entering the transmission state, which is equal to $(PW_{\mu p(a)} + PW_{RF-active} + PW_{tx-active}) \cdot 36\mu s \cdot N_r$, where $PW_{tx-active}$ represents the energy consumption of energizing the transmitter.
- ϵ_{52} represents the energy consumed by a tag listening for any incoming wake-up signals, which is equal to ϵ_{45}.
- ϵ_{55} represents the energy consumed for a tag to transmit feedback data. It can be calculated by $(PW_{\mu p(a)} + PW_{RF-active} + PW_{tx-active}) \cdot [(N_{B-resp} \cdot (L_{B-resp} - 1) + N_{p2p-resp} \cdot (L_{p2p-resp} - 1)) \cdot 36\mu s] \cdot N_r$

After applied the energy parameters, such as the power consumption of microprocessor, the energy matrix is given as:

$$E_{\mu p} = \begin{pmatrix} 206.250000 & 0.598950 & 0 & 0 & 0 \\ 0 & 0 & 23.740200 & 0 & 0 \\ 0 & 0.653400 & 16498.350000 & 0.653400 & 0 \\ 0.653400 & 0 & 0 & 8249.175000 & 5.405400 \\ 0 & 5.405400 & 0 & 0 & 59994.000000 \end{pmatrix} \tag{3.6}$$

Therefore, the energy probability matrix $EP_{\mu p}$ can be obtained by:

$$EP_{\mu p} = P_{\mu p} \times E_{\mu p}$$

and the State Energy Matrix, $SEM_{\mu p}$, can be obtained in Eq. (3.7).

$$SEM_{\mu p} = \left[\sum_{i=1}^{5} EP_{1i} \sum_{i=1}^{5} EP_{2i} \sum_{i=1}^{5} EP_{3i} \sum_{i=1}^{5} EP_{4i} \sum_{i=1}^{5} EP_{5i} \right]^{-1} \tag{3.7}$$

The total energy consumption, $E_{total-\mu p}$, for one active tag can be calculated by Eq. (3.8). This calculation of the energy consumption is based on one circumstance, such as an environment with 20 tags, 20 point-to-point commands, etc. More detailed results can be found in Section 3.4.3.

$$E_{total-\mu p} = \Pi_{\mu p} \cdot SEM_{\mu p} \tag{3.8}$$

3.4.2 Discrete Time Markov Energy Model for MP-μP-SB Model

Figure 24 illustrates the state diagram of the discrete time Markov process μP-SB model, which is the discrete time Markov process model used for modeling the energy consumption of a micro-controller and a *Smart Buffer* within one active RFID tag. The transistion matrix of the MP-μP-SB model can be obtained in a square array as $P_{\mu p+SB}$ in Eq. (3.9). This transition matrix is identical to the $P_{\mu p}$ of the discrete time Markov μP-SB model. The details of each transition probability can be found in Eq. (3.1) in Section 3.4.1.

$$P_{\mu p+SB} = \begin{pmatrix} p_{11} & p_{12} & 0 & 0 & 0 \\ 0 & 0 & p_{23} & 0 & 0 \\ 0 & p_{32} & p_{33} & p_{34} & 0 \\ p_{41} & 0 & 0 & p_{44} & p_{45} \\ 0 & P_{52} & 0 & 0 & p_{55} \end{pmatrix} \tag{3.9}$$

After applying the same environmental parameters, the transition matrix $P_{\mu p+SB}$ can be attained by Eq. (3.10). Thus, the stationary distribution can be achieved by $\Pi_{\mu p+SB} = \Pi_{\mu p+SB} \cdot P_{\mu p+SB}$.

$$P_{\mu p+SB} = \begin{pmatrix} 0.774011 & 0.225989 & 0 & 0 & 0 \\ 0 & 0 & 1.000000 & 0 & 0 \\ 0 & 0.007858 & 0.992102 & 0.000039 & 0 \\ 0.000039 & 0 & 0 & 0.992024 & 0.007937 \\ 0 & 0.007937 & 0 & 0 & 0.992063 \end{pmatrix} \tag{3.10}$$

The key difference between these two models, μP Model and μP+SB Model, is the energy matrix, $E_{\mu p+SB}$, which can be found in Eq. (3.11). With the assistance of the *Smart Buffer*, the

microprocessor remains in sleep mode and the *Smart Buffer* enters the active mode in *State 2* and *State 3* in Figure 24. The other difference is that the *Smart Buffer*, as well as the microprocessor, need to be concerned with in the energy calculation of each transition. Therefore, the energy elements in the energy matrix are redefined as following:

$$E_{\mu p+SB} = \begin{pmatrix} \epsilon_{11} & \epsilon_{12} & 0 & 0 & 0 \\ 0 & 0 & \epsilon_{23} & 0 & 0 \\ 0 & \epsilon_{32} & \epsilon_{33} & \epsilon_{34} & 0 \\ \epsilon_{41} & 0 & 0 & \epsilon_{44} & \epsilon_{45} \\ 0 & \epsilon_{52} & 0 & 0 & \epsilon_{55} \end{pmatrix} \tag{3.11}$$

- ϵ_{11} represents the energy consumed by a tag remaining in the sleep state. It can be obtained by $(PW_{\mu p(s)} + PW_{SB(s)} + PW_{RF-sleep}) \cdot (T_{hour} - T_t \cdot N_r)$, where $PW_{\mu p(s)}$ represents the power consumption of a microprocessor remaining in sleep mode, $PW_{SB(s)}$ indicates the power consumption of a *Smart Buffer* remaining in sleep mode, and $PW_{RF-sleep}$ represents the energy consumed when a RF front-end circuit remains in sleep mode.
- ϵ_{12} represents the energy consumed by a tag receiving a wake-up signal. It can be calculated by $(PW_{\mu p(s)} + PW_{SB(a)} + PW_{RF-sleep}) \cdot T_{wakeup} \cdot N_r$, where $PW_{SB(a)}$ indicates the power dissipated when a *Smart Buffer* circuit remains in active mode.
- ϵ_{23} represents the energy consumed by a tag receiving a preamble signal. It can be calculated by $(PW_{\mu p(s)} + PW_{SB(a)} + PW_{RF-active}) \cdot T_{preamble} \cdot N_r$
- ϵ_{32} represents the the energy consumption for a tag to realize the incoming command is non-relevant, which can be attained by $(PW_{\mu p(s)} + PW_{SB(a)} + PW_{RF-active}) \cdot 36\mu s \cdot N_r$.
- ϵ_{33} represents the energy consumed by a tag receiving an incoming command, which can be obtained by $(PW_{\mu p(s)}+PW_{SB(a)}+PW_{RF-active}) \cdot [(N_B \cdot (L_B-1)+N_{p2p} \cdot (L_{p2p}-1)) \cdot 36\mu s] \cdot N_r$, where L_B and L_{p2p} are the length of a one-to-all command and the length of a point-to-point command respectively.
- ϵ_{34} represents the energy consumed by a tag entering the processing-command state, which is equal to ϵ_{32}.
- ϵ_{41} represents the energy consumed by a tag to find out that the incoming command is a sleep command and put the microcontroller into sleep mode, which is equal to ϵ_{32}.

- ϵ_{44} represents the energy consumed by a tag generating feedback data. It can be calculated by $(PW_{\mu p(a)} + PW_{SB(a)} + PW_{RF-active}) \cdot [(N_{B-resp} \cdot (L_{B-resp} - 1) + N_{p2p-resp} \cdot (L_{p2p-resp} - 1)) \cdot 36\mu s] \cdot N_r$, where L_{B-resp} and $L_{p2p-resp}$ are the lengths of the response data for a one-to-all command and a point-to-point command respectively.
- ϵ_{45} represents the energy consumed by a tag entering the transmission state, which is equal to $(PW_{\mu p(a)} + PW_{SB(a)} + PW_{RF-active} + PW_{tx-active}) \cdot 36\mu s \cdot N_r$, where $PW_{tx-active}$ indicates the energy consumption of energizing the transmitter.
- ϵ_{52} represents the energy consumed by a tag to listen for any incoming wake-up signals, which is equal to $epsilon_{45}$.
- ϵ_{55} represents the energy consumed by a tag to transmit feedback data. It can be calculated by $(PW_{\mu p(a)} + PW_{SB(a)} + PW_{RF-active} + PW_{tx-active}) \cdot [(N_{B-resp} \cdot (L_{B-resp} - 1) + N_{p2p-resp} \cdot (L_{p2p-resp} - 1)) \cdot 36\mu s] \cdot N_r$

After applying the energy parameters, such as the power consumption of microprocessor, the energy matrix is given as:

$$E_{\mu p+SB} = \begin{pmatrix} 206.250000 & 0.002756 & 0 & 0 & 0 \\ 0 & 0 & 0.114450 & 0 & 0 \\ 0 & 0.013194 & 333.148500 & 0.013194 & 0 \\ 0.013194 & 0 & 0 & 8249.652400 & 5.841900 \\ 0 & 5.841900 & 0 & 0 & 59994.605000 \end{pmatrix} \tag{3.12}$$

Therefore, the energy probability matrix $EP_{\mu p+SB}$ can be obtained by:

$$EP_{\mu p+SB} = P_{\mu p+SB} \times E_{\mu p+SB}$$

and the State Energy Matrix, $SEM_{\mu p+SB}$, can be obtained in Eq. (3.13).

$$SEM_{\mu p+SB} = \left[\sum_{i=1}^{5} EP_{1i} \sum_{i=1}^{5} EP_{2i} \sum_{i=1}^{5} EP_{3i} \sum_{i=1}^{5} EP_{4i} \sum_{i=1}^{5} EP_{5i} \right]^{-1} \tag{3.13}$$

The total energy consumption, $E_{total-\mu p+SB}$, for one active tag can be calculated by Eq. (3.14). This calculation of the energy consumption is based on one circumstance, such as an environment with 20 tags, 20 point-to-point commands, etc. More detailed results can be found in Section 3.4.3.

$$E_{total-\mu p+SB} = \Pi_{\mu p+SB} \cdot SEM_{\mu p+SB} \tag{3.14}$$

3.4.3 Results

Based on these two models, an analysis of power dissipation is able to depict the impact of a *Smart Buffer* on an RFID tag based on different communication patterns such as broadcast and point-to-point communication. In three experimental scenarios, a general assumption is that only one RFID reader (R_0) starts the communication with a field of RFID tags (T), containing N tags, with a wake-up signal followed by a `collection` broadcast command. The reader then submits a series of k point-to-point commands. Finally, the reader puts all the tags to sleep using a point-to-point sleep command. The duration of each experimental scenario is one hour and each experiment is repeated 20 times with the average of all trials taken.

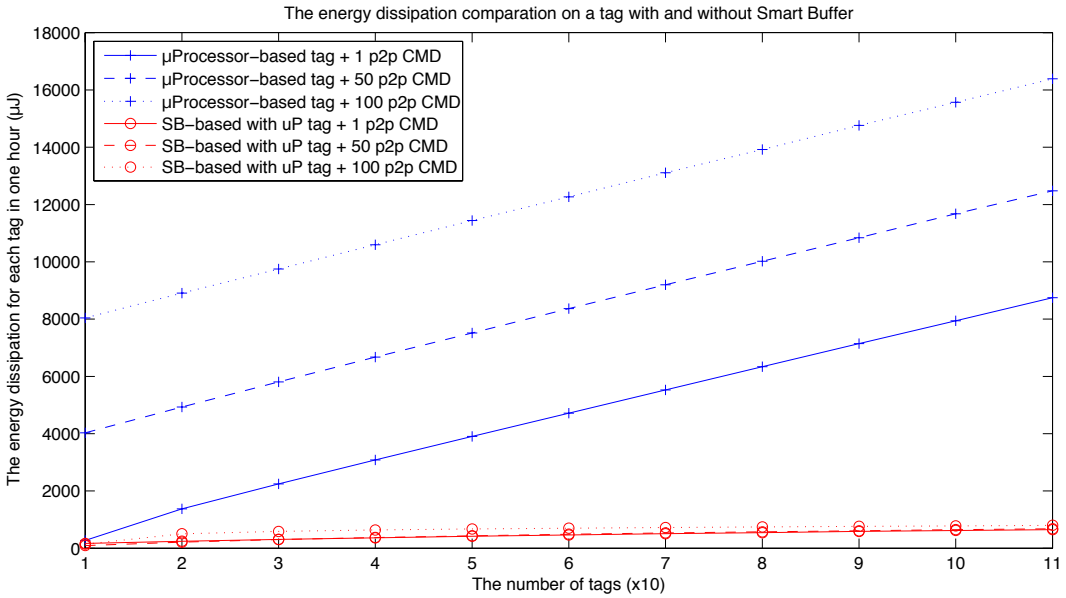


Figure 25: The energy impacts on a tag for point-to-point communication patterns

The results of three different scenarios are shown in Figure 25. Scenario one (solid line) sets $k = 1$. Scenario two (dashed line) sets $k = 50$. Scenario three (dotted line) sets $k = 100$. Data points with "+"s represents the μprocessor only tag, while data points with "o"s have a *Smart Buffer*. In this scenario the point-to-point command selected is the `set owner id` command. The average energy saved by adding the *Smart Buffer* is 82%, 88%, and 90% per tag where $N = 100$ for scenarios one, two, and three, respectively. As expected, as k increases, the energy consumed by the μprocessor-based tags increases because the μprocessor is processing every packet. In contrast, the smart buffer tags consume approximately the same energy regardless of the number of messages because the μprocessor is rarely activated. The increase in energy consumed by the μprocessor tags as N increases is due to the point-to-point sleep commands which scales with N. The *Smart Buffer* based tags avoid this increase.

3.5 CONCLUSION

This chapter discussed energy reduction issues. Two observations are made that energy is wasted mainly when (1) irrelevant point-to-point commands are sent to a group of tags, and when (2) the receiver of each tag always listens to the incoming signals. Based on these observations, the *Smart Buffer* technique is proposed to assist power saving for a passive active RFID tags (PARTs). The *Smart Buffer* is capable of filtering irrelevant commands and remaining the microcontroller in sleep mode until it is addressed. Our experiments show that the average energy saved by adding the *Smart Buffer* is 82%, 88%, and 90% per tag where $N = 100$ for scenarios one, two, and three, respectively Section 3.4.3.

The amount of power saved by using the *Smart Buffer* is highly dependent on the scenario, especially for a point-to-point scenario. First, the *Smart Buffer* must consume less power than the active tag controller. This is supported by the data shown in Section 3.3.10. Secondly, the amount of power saved depends on how many successful accesses the tag receives on average, how long the processor is active, etc. We examine several scenarios and the *Smart Buffer* impact in Section 3.4.3. As seen in Section 3.3.10 the power savings seen from using the *Smart Buffer* can be several orders of magnitude for the microprocessor based tag.

4.0 MULTI-LAYER SECURITY ARCHITECTURE

4.1 INTRODUCTION

Applications for RFID continue to expand into domains such as electronic passports, electronic payment systems, and electronic container seals. These applications have a risk of unauthorized access to sensitive biometric or financial information through the RFID tag or tag communication.

However, as RFID devices are intended to be small and relatively simple devices, security protocols and techniques can significantly lag behind the other details such as correctness, read rate, power consumption, etc. As such, the state of security in RFID systems is generally weak compared to other mature computational technologies such as Internet servers, shared computing workstations, and even smart cards.

RFID systems require security features to be implemented using techniques that provide a high strength of protection while not significantly increasing the complexity of the system as complexity can increase power consumption and implementation cost. Thus, it is important to develop new security techniques that take advantage of the fundamental properties of RFID communication such as physical layer protocols, low-power communication extensions, sleep modes, physical implementation variations, etc.

This chapter provides a survey of existing security techniques employed in RFID systems including authentication and encryption as well as commonly employed types of attacks, and security features of existing RFID standards. This discussion is contained in Section 4.2. Subsequently, a security scheme, as also called *layers of security*, for an RFID tag that combines features from passive and active tags is described in Section 4.3. Our security system takes advantage of the

fundamental properties of the tag to create multiple levels of low-complexity security features, that when combined provide a strong protection. Some concluding remarks are presented in Section 4.4.

4.2 SECURITY IN RFID SYSTEMS

RFID systems in general and RFID tags in particular have the requirement of a power versus area tradeoff. The power consumed by the tags impacts the range of the device (passive) or lifetime of the tag (active) where the area dictates the cost of the device. In general, the industry has focused more on the area constraint than the power constraint. The area optimized device for reduction in cost. The composed of less than 10,000 gates.

Security concerns have often been a secondary concern for vendors primarily because strong authentication and encryption algorithms are complex and would significantly increase the cost and power budget of the tag. Sarma *et al.* discusses the security risks and challenges of low cost RFID tags [38]. For example, in the security. In contrast, commercial implementations of the advanced encryption standard require approximately 20,000 - 30,000 gates.

As a result, security techniques for RFID tags must leverage specific details of RFID communication to create low-overhead secure transmissions. It is important to ensure that neither the tag and reader are malicious with the intent to access restricted data or destroy data. To prevent this, authentication techniques for RFID are described in Section 4.2.2. Additionally, once a reader and tag have entered into a trusted communication, it is important to prevent a malicious device from overhearing the communication and stealing data. To prevent this, encryption techniques for RFID are described in Section 4.2.3.

4.2.1 Types of Attacks for RFID Systems

An intrusion or an attack to a computer on a traditional network has a physical limitation due to physical network links. An intruder attempting to perform an attack requires either some physical connection to the network through tapping a network cable or access to some terminal on the

victim's network. Those attacks may be traceable by detecting traffic over particular network links or tracing with IP addresses, etc. However, a malicious access to computing nodes in an open RF environment, such as RFID network, wireless sensor network, or other wireless or RF network is not easy to be detected and much more difficult to track. For example, data travels over the air which is easily detectable by many non-trusted devices. As a result, there is a high demand to increase the security and privacy for RFID networks. To understand potential attacks to RFID networks we have summarized several common attacks in the following subsections.

4.2.1.1 Eavesdropping An unauthorized observer attempts to capture the exchanged information between the reader and tags without permission. This is a passive style attack. RFID communication occurs over the air and is thus an easy target for an eavesdropper to monitor data transmitted without being detected. As a result, sensitive corporate or personal information may be captured, recorded and analyzed by a malicious attacker. Complex data encryption and decryption algorithms would be suggested as good approaches to prevent this type of attack. Unfortunately, complex data encryption such as AES encryption is often too expensive for low cost and/or passive RFID tags.

4.2.1.2 Spoofing In the spoofing attack, a malicious device forges an existing tag with a cloned tag that can interact in a trusted manner by a reader. For example, after employing the eavesdropping attack, an unauthorized third party may be able to analyze and reverse engineer the captured encrypted information. Often, this information is easy to crack as the encryption is typically simple due to the limited design complexity of these tags. After the attacker breaks the architecture or the secret key of the encryption algorithm, they are capable of cloning the tag to deceive a legitimate reader.

This type of attack is able to defeat access control [39], allow unauthorized access to personal property and systems such as e-payment [40]. Mutual authentication and complex data encryption are suggested for preventing this type of attack. As before complex data encryption is often too expensive for many RFID systems. When adding mutual authentication as a requirement, the cost is increasingly prohibitive.

4.2.1.3 Denial of service

In computer networks, a denial-of-service (DoS) attack attempts to paralyze computers so that they are not available for a legitimate access. One DoS attack is to flood a network in order to disrupt services to legitimate users or occupy the most of the network bandwidth. `SYN` flood and `PING` flood are two well known DoS attacks in computer networks.

From the RFID system perspective, a denial-of-service attack may cause either RFID readers or RFID tags to malfunction [41]. In consequence, it will cause a tag temporary or permanent paralysis where the tag becomes unreadable or untraceable. This type of attack may cause problems to applications such as automated inventory processing (e.g. shopping) or tracking military shipments.

The denial-of-service attack can be easily performed on active RFID tags where tags utilize batteries. The energy of battery will be exhausted at a drastic rate when an attacker surreptitiously transmit a large number of RFID commands.

4.2.1.4 Brute force

A brute force attack is an approach to guess a secret key of an encryption algorithm or to gain an access of a computer resource by exhaustively trying all possible keys. For example, in order to search the 56-bit secret key of a DES encryption system, a hacker needs to try 256 possible keys, meaning 255 trials (36x1015 trials), to find the match. To exhaustively compute this is easily accomplished within approximately seven days as illustrated by the study of Kumar *et al.* [42].

Researchers at the Johns Hopkins University Information Security Institute and RSA Laboratories (JHU-RSA) successfully demonstrated the vulnerability of a 40-bit RFID-enabled cryptographic tag [40]. The Digital Signature Transponder (DST) by Texas Instruments, utilizes a 20-round unbalanced Feistel cipher encryption algorithm with 40-bit secret key [40]. This is the RFID technology behind ExxonMobil speedpass, a wireless payment system tied into a credit card. The research group at JHU-RSA built a parallel Field Programmable Gate Array (FPGA) platform to perform a brute force attack to crack the 40-bit secret key. By utilizing parallel computation of 16 FPGA devices, the 40-bit secret key can be recovered on average in less than one hour. This group suggests that a 40-bit cipher key is vulnerable for brute-force attack and a 128-bit cryptographic algorithm is recommended for commercial and homeland security applications.

4.2.1.5 Relay The relay attack, also called the "man in the middle" attack, is an example of a spoofing attack where a third party deceives the first legitimate party thinking he is the second legitimate party and deceives the second party into thinking he is the first party. In effect the third party relays the message from one party to the second and vice versa. Thus, the two authorized parties intending to communicate with each other are tricked into a three party conversation without being aware of the third party. Two relay-attack examples are shown in [43, 44].

The longest distance to make this type of attack is approximately 50 m [44]. However, the relay attack is limited by the distance between the reader and the tag. For example, as the distance between the tag and reader increases, the time of relaying signals by the third party between the reader and the tag increases as well. According to the ISO 18000 part 6C standard, the reader waits for the tag response within 77 μs after transmitting a `Query` to the tag. Otherwise, the reader will terminate this communication attempt. Therefore, a possible prevention for this type of attack is to set a more strict time requirement to prevent this attack even at reasonably close distances.

4.2.1.6 Side channel Unlike previous attack types, in a side channel attack a cryptanalyst exploits some information measured externally from the device [45], such as timing [46], power [47], or electromagnetic [48, 49] details. Side channel attacks based on power analysis have seen particular interest for RFID and similar systems.

Power analysis techniques measure or monitor power consumption of a cryptosystem by physically probing a ground pin [47]. These analysis techniques have been classified into two sets: simple power analysis (SPA) and differential power analysis (DPA) [47]. The accuracy of performing SPA can be easily affected by the signal-noise-ratio (SNR) on the probed pin. However, the DPA technique not only measures power consumption of a cryptosystem, but also requires a statistical analysis based the collected power consumption information. Thus, it can minimize the effect of SNR because it differentiates output power traces.

The statistical technique of DPA works using a *correlation function* that indicates the correlation between using a guessed key and the unknown secret key of the cryptosystem. The key is subdivided into sub keys and each permutation of the sub key is tested while the remainder of the key is held constant. If the correct guessed sub key is used, the correlation function will indicate high correlation (*e.g.* spikes of differential power traces) between the correct guessed sub key and

the actual secret sub key. At the end the entire correct key is composed of the high correlation sub keys. By using this technique the number of keys to try is reduced by orders of magnitude over a brute force attack.

In 1999, Kocher *et al.* announced a DPA attack against a Data Encryption Standard (DES) co-processor [47]. This work describes DPA traces with correct and incorrect guessed sub keys against a secret key of the DES co-processor and shows the actual power spikes when the correct sub key is guessed. DPA attacks have been proven effective on various cryptosystems such as an AES ASIC co-processor [50], AES hardware-based implementations [51, 52], and a DES FPGA implementation [53]. To increase resistances of DPA, many protection countermeasures are proposed [54, 51, 55].

For RFID systems, Rakers *et al.* describes a DPA attack on a ISO 14443 RFID enabled "contactless" smartcard [56]. In 2002, Messerges *et al.* [57] illustrate an implementation of a similar DPA attack on smartcards. Rakers also describes an approach to protect the contactless smart card from DPA attacks by exploiting an isolation circuit. An isolation circuit is primarily used on an ASIC to prevent bit error rate (BER) degradations due to digital interference. It behaves like a current source which is independent of power consumption of the digital circuitry. Thus, the power signature of the digital circuitry is reduced by a factor of 2000 or 66dB, which significantly increases the difficulty to perform DPA.

4.2.2 Authentication Techniques in RFID Systems

Authentication is a mechanism or process to either verify the identification of a party with which communication occurs or to verify the integrity of received messages without their having been modified. For example, symmetric encryption mechanism provides a form of authentication among those who share the same secret key. In addition, asymmetric encryption provide authentication and confidentiality among those who share public keys.

An alternative approach to authenticate a message is the use of a hash function. The hash function maps a variable-length message into a fixed-length hash value. It is easy to convert a

message into a hash value, but it is difficult to recover the message from a hash value without appropriate information. An advanced hash function, such as Secure Hash Algorithm (SHA), combines hash function with secret keys.

4.2.2.1 Lightweight Authentication Protocols Vajda and Buttyan [58] describe several lightweight authentication protocols for RFID tags including encrypting a challenge message using XOR, subset, square, RSA, and KNAPSACK. For example, in the XOR authentication protocol, a reader and a tag initially share two different, independent random keys, k_R^0 and k_T^0 respectively. The RFID reader initiates the authentication protocol, shown in Eq. (4.1), by transmitting a initial challenge message of x^0 XOR k_R^0 to the tag. The tag retrieves the challenge message by computing XOR with k_R^0 and responds with another challenge message computed by XOR with k_T^0 to the reader. The reader extracts x^0 and compares with the one it transmitted to the tag.

$$R \rightarrow T : x^0 \oplus k_R^0 \tag{4.1}$$
$$T \rightarrow R : x^0 \oplus k_T^0$$

For subsequent runs of the protocol, x^i and k_R^i are random numbers generated by the reader. The reader transmits not only the encrypted challenge message but also the k_R^i XOR k_R^{i-1}. Eq. (4.2) details the protocol for XOR. The other four authentication protocols utilize a similar flow but apply additionally complex encryption methodologies to encrypt challenge messages.

$$R \rightarrow T : x^i \oplus k_R^i || k_R^i \oplus k_R^{i-1} \tag{4.2}$$
$$T \rightarrow R : x^i \oplus k_T^0$$

Peris-Lopez *et al.* propose a new lightweight mutual authentication protocol (LMAP) technique targeting area optimized implementation for low cost devices such as RFID tags [59]. The LMAP protocols is decomposed into four steps: tag identification, mutual authentication, index updating, and key updating. The implementation of their proposed protocol needs 86, 173, 346, 691, and 1037 gates for 8-, 16-, 32-, 64-, and 96-bit authentication messages, respectively.

Bernardi *et al.* propose a new architecture of a UHF RFID transponder compatible with the ISO 18000 part 6C standard and extending the system with secure authentications. Additionally,

they proposed a multi-layer network protocol for ISO 18000 part 6C. The architecture includes a microprocessor, output control unit, memory blocks, I/O encoder and decoder, and an encryption module. A simplified asymmetric RSA encryption algorithm is implemented in the encryption module, which encrypts 16-bit input plain-text with a 1024-bit fixed secret key and provides 1024-bits of encrypted data. For a 90-nm HMOS process, a design running at 1.9 MHz required 1.4 mm^2 of area and consumed approximately 1.5 mW of power.

Bolotnyy and Robin proposed a hardware-based authentication approach based on physically unclonable functions (PUFs) [60]. PUFs take advantages of variations in silicon processes and operating conditions to increase authentication strength. The PUF is based on a silicon random number generator [61, 62, 63]. The value generated from the silicon random number generator depends highly on the wire delay, temperature, thermal gradients, etc of the physical device. The proposed RFID architecture combines the PUF with a universal hash function [64]. The proposed 64 PUF hash function circuit is composed of 64 single bit units and one oscillating counter circuit. Each single bit unit requires an estimated 8 gates and the oscillating counter circuit are estimated at 33 gates, which when extrapolated to a 64-bit PUF hash function requires 545 gates.

Leung *et al.* proposed the use of a cryptographic nonce as the tag identifier designed to avoid the tracing and cloning style attacks [65]. A cryptographic nonce is a one time use number. Because the identifiers are used only once, it is not possible to trace the tag with its identifier unless the sequence of identifiers is known to the attacker. Similarly, it is not possible to clone the device. However, it is relatively easy for the tag and reader to be unsynchronized through a denial of service attack.

4.2.2.2 Symmetric authentication protocols A symmetric authentication protocol is an authentication mechanism for those who share the same secret key to verify authenticated messages.

In describing the tradeoffs of security versus cost of RFID tags, Weis *et al.* proposed hash-lock and randomized hash-lock authentication protocols [66, 67]. As hash-lock authentication uses hash-based access control using one-way hash functions. By comparing an internal "metaID" with the value of a hash function of a received key from the reader, the tag unlocks itself to the reader upon a match.

The randomized hash-lock authentication protocol [66, 67] extends the hash-lock technique to include randomly generated numbers as identifiers in the exchange. For example, a tag is equipped with a random number generator. When the tag is queried it responds with a pair $(r, h(ID_k||r))$ where r is a random number, h is a hash function, and $ID_k||r$ is the kth ID concatenated with the random number. The reader performs a hash function of all ID in the database concatenated with r until it finds a match. To unlock the tag, the reader transfers ID_k back to the tag.

4.2.2.3 Asymmetric authentication protocols An asymmetric authentication protocol utilizes a public and private key to perform the authentication process. For example, the reader transmits an authentication message to the tag encrypted by the tag's public key. The tag decrypts the message with its private key. The tag responds with the same authentication message back to the reader encrypted by the reader's public key. The reader then decrypts it with its private key and checks it against the message it initially sent. General asymmetric encryption implementations such as Elliptic Curve Cryptography (ECC) are exponential computations inefficient for pervasive computing environments [68].

Niederreiter asymmetric encryption is an existing asymmetric encryption algorithm that does not require exponential complexity but requires matrix operations [69]. Cui *et al.* introduced a lightweight asymmetric authentication protocol designed for RFID devices [68]. This technique employs the Niederreiter asymmetric encryption technique to perform a challenge response authentication protocol by using public and private keys. The public and private keys are built in Niederreiter encryption algorithm by generating several matrices and an (*n,k*)-linear code.

4.2.2.4 Strong authentication protocols Feldhofer proposed a strong authentication technique for RFID based on the Advanced Encryption Standard (AES) [12]. First the reader generates a random number N_{reader} and sends it to the tag. The tag returns a random number N_{tag} and both the N_{tag} and N_{reader} encrypted with the AES secret key. The reader decrypts and verifies the encrypted N_{reader} is the same as originally sent. It then reverses the order of N_{reader} and N_{tag} and returns this, encrypted, back to the tag. If the reader decrypts the correct N_{tag}, then the reader and tag continue with their transaction.

For this implementation the system for AES encryption is an 8-bit co-processor for and thus requires only 3,600 gates. However, it requires approximately 1000 clock cycles to accomplish each encryption operation thus requiring approximately 10 ms to computation time. From a power consumption perspective, this implementation of the design utilizes 0.35 μm technology and requires 8.15 μA at 100 kHz.

4.2.3 Encryption Techniques in RFID Systems

Authentication provides a procedure to verify the identity of the parties to communication. In the case of RFID, this includes tags and readers. However, after the authentication has been completed, the readers and tags transmit insecure information. As the communication is wireless, it is very easy for a snooping device to pick up and steal this information. In many cases this information may be of a sensitive nature, such as biometric data, identification numbers such as credit card or social security numbers, corporate infrastructure details, etc.

In order to secure the information exchanged between readers and tags, data encryption is needed to protect these systems. However, current commercial data encryption modules may be too expensive in terms of area (cost) and power (range/lifetime) to be feasible for an RFID system. Therefore, several efforts have studied how to find "low-cost" encryption techniques suitable for RFID systems.

4.2.3.1 Symmetric key encryption A symmetric key encryption mechanism utilizes a single private key shared amongst readers and tags to encrypt or decrypt a message or data. The length of the key controls the level of security for this encryption quantified by the time taken to break the encryption with a brute-force attack. The length of key also correlates with the implementation complexity of the security technique. For example, a modern 128-bit symmetric-key encryption core costs more than 100,000 gates in area [70] while performing high throughput of encrypted data with the strength of 2^{128} possible key values. However, for low-cost RFID systems, an RFID tag has much stricter area constraint making this algorithm infeasible. Thus, implementations for RFID systems must trade off encryption strength against implementation complexity.

The Tiny Encryption Algorithm (TEA) [71] has been proposed for low-cost RFID data encryption [72]. A 32-bit TEA algorithm has been implemented and synthesized for a 0.35 μm CMOS technology. The implementation required 0.21 mm^2 of area and could achieve a maximum clock frequency of 50 MHz. The dynamic power dissipation is estimated at 7.37 μW with a 25.6 kHz clock.

The TinyAES architecture is based on AES-128 for RFID systems [12]. The data path of the hardware-based architecture is 8-bits wide. The silicon implementation result shows that the TinyAES module draws a current of 3.0 μA and consumed the power of 4.5 μW when operated at 100 kHz and 1.5V. The core needs an area of 0.25 mm^2 on a 0.35 μm CMOS technology, which compares roughly to 4400 gates.

Another approach describes a tradeoff between area and timing of an AES encryption implementation [73]. This implementation was introduced to reduce timing requirement of AES encryption by optimizing the the data processing of the algorithm. The 8-bit hardware-based architecture implemented in 0.25μm CMOS technology has a gate count of 3,868 gates and requires 870 clock cycles at 10MHz. However, the power dissipation information for this implementation is not provided.

4.2.3.2 Asymmetric key encryption An asymmetric encryption algorithm relies on one key, called a public key, for encryption and a different, but related key, called a private key for decryption [74]. Thus, to send an encrypted message, first the receiver provides the sender with its public key. The sender encrypts the message with the public key and the receiver decrypts the message with its private key. It is important not to be able to determine the private key from the public key. To increase security by adding an authentication component, to verify that the message came from a trusted sender it would be required to decrypt the message with the receiver's private key and sender's public key.

Asymmetric key encryption requires more complex algorithms compared to symmetric key encryption for comparable encryption strengths, but also provides many advantages such as not having to divulge and share a secret key and the capability to manage the system for individual devices and groups of devices. For RFID systems, implementing an asymmetric encryption scheme is a significant challenge.

Elliptic Curve Cryptography (ECC) is a type of public-key cryptography proposed for low cost RFID systems [75] and in particular for ISO 18000 Part 6C [76]. The ECC processor is composed of a control unit, arithmetic unit (ALU), and memory (RAM and ROM). The ECC synthesized in 0.25 μm CMOS technology shows that the ALU requires 6300 - 7800 gates depending on the size of the Galois Field adder GF(2^m) where m is the length of the key. It can also be much larger depending on the size of the memory block.

The study of Leung *et al.* [65] describes an implementation of a 173-bit ECC crypto-processor based on an Optimal Normal Basis (ONB) methodology. This implementation is designed for a low-power inductive RFID application such as the ISO 14443 contactless smartcard standard, which operates at 13.56MHz. The 173-bit ECC crypto-processor is capable of operating at 18MHz and executing an ECC encryption process within 7.56 ms and requires 95mW with a 3.3V power supply.

4.2.4 Current Security in RFID Standards

4.2.4.1 ISO 18000 Part 7 The ISO 18000 Part 7 standard [3] is the most popular protocol for active ultra high frequency (UHF) systems. To prevent malicious access to the tags, a password style authentication mechanism is provided in the standard by the `set password`, `set password protect`, and `unlock` commands. The `set password` command sets an internal password required for further tag accesses. The `set password protect` command enables or disables password authentication. The `unlock` command allows unprotected access to the tag. Unfortunately, password protection is not a strong authentication technique, and passwords can easily be stolen and spoofed.

4.2.4.2 ISO 18000 Part 6C The ISO 18000 Part 6C standard [22] is a recent adoption by ISO of the Class 1 Generation 2 "Gen 2" RFID specification [23] for passive UHF tags from electronic product code (EPC) global, Inc. Even prior to its standardization by ISO, Gen 2 tags have become very popular and widely used for applications requiring passive RFID.

Neither the ISO standard or the Gen 2 specification contain any particular requirements for security in compliant tag or reader implementations and typical implementations employ minimal security features.

The interrogator or reader can lock or unlock each individual area of memory. This includes access to the access or kill passwords, the electronic product code (EPC) memory bank, or tag identification (TID) memory bank. When the tag is locked, the passwords cannot be read or modified and all other memory banks are write protected. Additionally, the reader can *permalock* the lock status for a password or memory bank so that it is not changeable. As the name implies, one permalock is asserted to a particular memory block, it cannot be changed. Finally, the tag can be permanently deactivated if the tag receives the `kill` command with the correct password. After the tag is killed it no longer responds to interrogator commands. Interestingly, killed tags are actually alive and can still receive passive power and actually buffer incoming packets, but never respond.

Jeuls proposes an authentication protocol to combat tag cloning even in environments with untrusted readers [77]. This protocol assumes an authenticated tag that contains its own unique EPC identifier and its own 32-bit secret key. The readers authenticated to communicate with the tag contain a database of secret keys associated with the authenticated tags. During a transaction, a tag identifies itself and the reader verifies the identifier with its database. The reader responds with the appropriate secret key. Finally, the tag verifies the secret key from the authenticated reader. Unfortunately, if any adversary eavesdrops during this authentication process the tag identifier and secret key can be captured. Data encryption is required to protect this technique for this information leakage.

4.2.4.3 ISO 14443 The ISO 14443 standard [78] is the most popular standard for high frequency proximity cards or what are also called contactless RFID cards or contactless smartcards shaped at the size of a credit card. ISO 14443 compliant devices typically have a range of approximately 3 inches and cost less than $5. The devices are passively powered. The ISO 14443 standard does not specify any specific cryptographic algorithm or authentication protocol. However, security features, such as authentication and encryption, available in the smart card standard ISO 7816 [79] are also compatible for ISO 14443 devices [80]. The ISO 7816 smartcards, which

are in contact with the readers when they are accessed, can be directly powered. Thus, the passively powered ISO 14443 devices have a significantly lower power budget, making many of these cryptographic algorithms difficult to implement.

Mandel *et al.* describe how to use a $30 device to perform a relay attack on proximity cards that adhere to the ISO 14443 standard [39]. The technique takes advantage of the passive authentication feature employed by these devices rather than more secure alternatives such as active authentication and a challenge-response protocol. The broadcast signal from the proximity card is captured and then relayed using a $30 AM transmitter to a gate that subsequently opens. In addition to being able to recreate the signal, after snooping several proximity ID and broadcast signals, the passive authentication structure in the card was easily reverse engineered.

Kfir and Wool analyzed the man-in-the-middle attack approach to perform relay attacking on the RFID-enabled contactless smartcard [43]. In their approach they use a theoretical model to describe the behavior of the device and show that it is possible to perform relay attacks when the authorized tag and reader are in relatively close proximity to each other.

Hancke implemented prototypes called a "Mole" and "Proxy" system to perform a practical relay attack against an ISO 14443 type-A contactless smartcard [44]. In this experiment, the distance of the relay attack is shown to be as far as 50 m.

The MiFare microprocessor [81] from Philips, a leading manufacturer of ISO 14443 chips, contains a crypto-co-processor capable of processing 3DES algorithms in hardware. 3DES is a variant of the data encryption standard encryption algorithm (DES) originally proposed by IBM [82] that avoids brute force attack vulnerabilities of DES and meet-in-the-middle vulnerabilities of double DES.

4.2.4.4 ICAO extension to ISO 14443 Personal information contained within a travel document, credit card, driver's license, or other identification is often extremely sensitive and results in concerns about information privacy. One of the applications that employs the ISO 14443 standard is electronic passports (E-passports). As this application raises a significant privacy concern, efforts have been made to add additional security to ISO 14443 specifically for this application.

Juels *et al.* employ basic access control to prevent the RFID chip from being accessed by an unfamiliar reader unless the reader can prove it is authorized [83]. To accomplish this, they employ

a digital signature cryptographic technique based on public key cryptographic algorithms [74]. However, as these devices are passively powered, the power budget for operating the tag is limited, which also limits the strength of authentication that can be employed. In this implementation the key length is very small making it relatively easy to defeat. Additionally, the key is hard wired into the device making it quickly obsolete.

E-passport cryptography is further addressed in the Machine Readable Travel Documents (MRTD) specification published by the International Civil Aviation Organization [84]. The ICAO requires a baseline security technique of passive authentication based on RSA, DSA, or a similar variant. It also goes further to describe techniques for active authentication, basic access control, extended access control, and data encryption.

The passive and active authentication measures ensure that the content of the RFID chip in the E-passport has not been modified or that the RFID tag has been improperly replaced or forged. Extended access control is a protection protocol to prevent unauthorized access to biometric information. Data encryption prevents against a third party listening to the transmitted biometric information.

4.2.4.5 ISO 18185 [85] The ISO 18185 standard [85] describes RFID tags that can be used for electronic container seals. Electronic container seals allow the tracking of a container all the way through a supply chain to verify that the items have not been opened, or tampered with. When the container is initially filled with cargo and sealed the ISO 18185 compliant tag controls a locking bolt which is put in place. After the container has been sealed it may not be resealed. Once the container reaches its destination it is opened by breaking the bolt and the RFID tag moves to the "opened" state, where it remains and cannot be changed. After use the tag is disposed of.

The container seal tags contain no significant security features, data, such as cargo contents, are transmitted in clear text. The only form of authentication is use of a unique tag identifier to avoid replication. The 18185 tag is susceptible to a snooping and spoofing attack. For example, an attacker can listen to a tag access to get the unique information. Then they can break the seal, and replace it with a spoofed tag that uses the same unique id as the tag placed prior to tampering.

4.2.4.6 ANSI NCITS 256 The American National Standards Institute (ANSI) published a standard for active RFID systems (ANSI NCITS 256) [4]. It defines an equivalent air interface characteristics and specifications to the ISO 18000 part 7 standard including the same preamble signal and data format. The differences from ISO 18000 part 7 deal with the types of RFID commands. Except the collection command, there are 13 data commands in ANSI 256 standard. However, none of them is used for access control, authentication or data protection. ANSI NCITS 256 is vulnerable to eavesdropping, tag cloning, data disclosure, and other standard attacks.

4.3 LAYERS OF SECURITY

The approach for securing RFID transactions is to employ security at multiple levels during the RFID transaction. These levels are applied to different layers in the communication scheme similar to the layers in the Open System Interconnection (OSI) Model. As such, the system assumes as the baseline architecture the passive active RFID tag (PART) [31] described in Section 2.2 and the use of a design automation technique to program the architecture [37, 36, 86] described in Section 4.3.3. Thus, the approach provides layers of security in the final RFID system including: (1) the passive activation layer (burst switch), (2) the active communication encoding (physical layer) (3) the use of encrypted data transmission (specified with the RFID design automation) and (4) physical security protection.

The passive activation layer is encoded to prevent a malicious reader from continuously activating the tag to drain its battery or to attempt to issue malicious commands. This layer secures and protects the security of the "Passive Transceiver Switch" in Figure 26. The physical layer is encrypted using a mixture of Manchester and Differential Manchester encoding. This layer of protection is developed at the level of the "*Smart Buffer*" in Figure 26. Finally, the encrypted data transmission and physical security protection are developed for the layer of "Controller" in Figure 26.

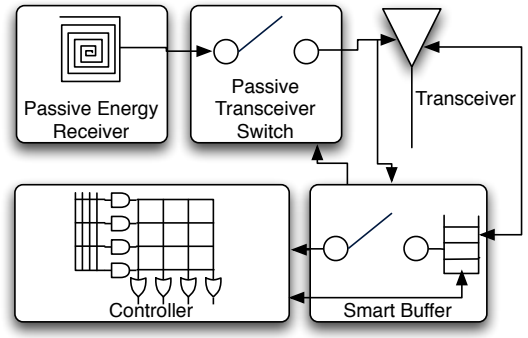


Figure 26: Overview of the ultra low-power active RFID tag.

4.3.1 Passive Activation Layer Security

The burst switch alone is not sufficient to replace the active receiver for RFID tags. The simple presence of RF energy such as noise or a communication with other wireless systems or even a malicious reader can cause the tag to wake up the active transceiver. To solve this problem, we have developed a signal encoding methodology using both hardware and software prototypes. The RFID reader generates bursts of energy of different durations like those shown in Figure 27. In the example from Figure 27, the reader generates four pulses with lengths of 2, 12, 3, and 9 time units. The tag must detect a unique code from these bursts in order to activate the remainder of the tag. The software based system is implemented with a PIC microprocessor [87]. The hardware based system is designed for implementation in an ASIC or SoC. The strength of the encoding is related to two components: the number of bursts in the sequence n and the unique number of different burst lengths detectable by the receiver b. Thus, the resulting number of unique codes is $n\dot{b}$. The main components of the detection circuit, shown in Figure 28 are two counters and a comparator. The first counter detects the value of the burst determined by the burst length requiring $\lceil \lg b \rceil$ bits and the second to tracks which burst is being checked in the sequence requiring $\lceil \lg n \rceil$ bits.

The clock speed of the circuit depends on the detection precision of the burst switch. For example, if we consider the detection of a 6 μs burst with a granularity of 1 μs we clock our circuit

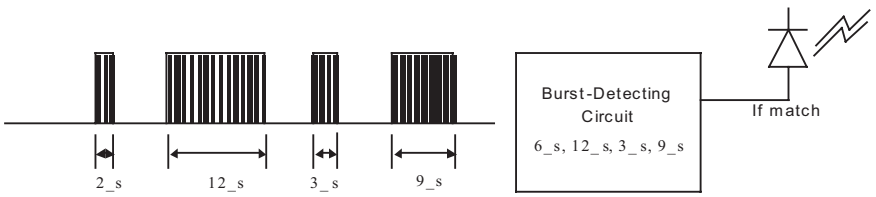


Figure 27: Example encoding for the burst switch. The wake-up signal contains four bursts of 2, 12, 3, and 9 time units, respectively.

at 10 MHz. However, as shown in Figure 29 with the solid line, a 6 μs burst was detected with non-zero probability for bursts ranging from 5.1 - 6.9 μs. This can be corrected by oversampling of 10X (e.g. clocking at 100 MHz instead of 10). Unfortunately, increasing the clock speed by 10X also increases the power consumed dramatically. However, only at 6.0 μs does detection occur with 100% probability.

Thus, rather than oversample, we consider the impact of increasing the number of pulses in the sequence. The heavy dashed line from Figure 29 shows the impact of placing four bursts together with the same deviations. Interestingly, the false positives drop off much more quickly than predicted (dotted line), which still predicts as high as 40% false positives after the actual number of false positives have already dropped to zero. By including a small amount of oversampling (2-3X) and breaking the burst detection into multiple bursts, it is possible to see negligible false positives and keep our power budget at a minimum.

Once the proper burst switch encoding has been received, the active transceiver is wakened to begin the active stage of communication. The encoded burst switch prevents a malicious reader from executing a denial of service attack on the passive-active tag to dramatically drain the battery.

4.3.1.1 Results To determine the minimum time increment for differentiating a different length pulse, we prototyped the digital portion of the hardware with Spartan 3 FPGAs and connected the generator and detector with a wire. Figure 29 shows the percent of reads that were detected as 6 μs for values ranging from 5 - 7 μs. In this experiment the deviation was approximately 1 μs. By considering four pulses, this deviation drops to 0% for 0.3 μs. When testing with the Lynx

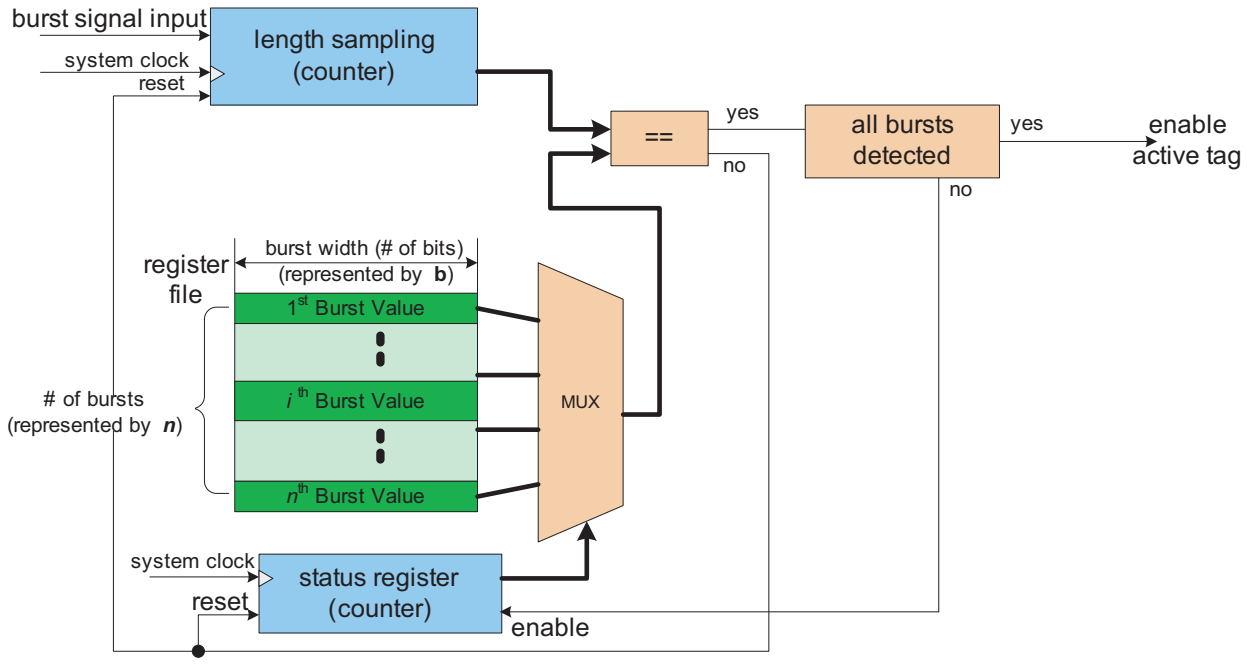


Figure 28: Architecture of the burst sequence detection circuit.

transmitter and receiver we found that the clock speed should be reduced significantly below 1 MHz as the resolution of the transceiver is at least an order of magnitude (100 kHz) slower [88].

The burst switch detector was implemented in ASIC hardware using Synopsys Design Compiler targeting 0.16 μm Oki cells and examined for power consumption using Synopsys PrimePower at 10 MHz, 1 MHz, and 100 kHz system clock rates. The number of unique representations for each burst was represented using 2, 4, 8, 16, and 32-bits and for each experiment the number of bursts was held constant at four. The area and power results are shown in Table 8. For comparison, the same functionality was implemented in a Microchip PIC12F635 8-bit ultra low power microprocessor requiring 200 μW of power [87]. Even operating at the highest clock speed 10 MHz, the power consumption for the largest design is less than 30% of the PIC. However, a 100 kHz clock speed design, closer to matching the capability of the transceiver, requires 300X less power than the PIC when processing. Interestingly, at such low clock speeds, the clock gated circuit provides little power advantage.

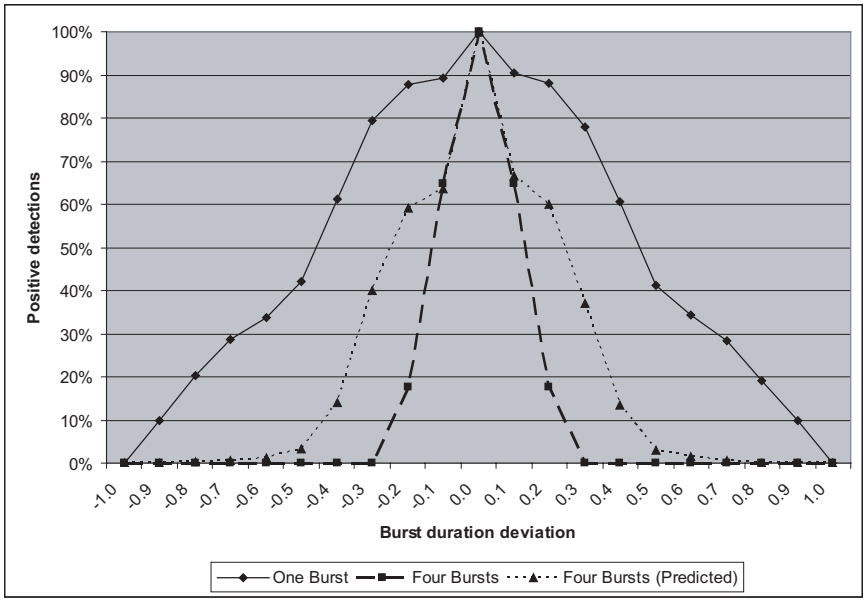


Figure 29: Percentage of false positives for one burst and four bursts detection between +/- 1 μs and granularity of 0.1 μs.

4.3.2 Physical Layer Security

Active RFID tags generally communicate using some form of Manchester encoding. Manchester encoding combines data communication with a synchronization clock. Each bit is contained within a *window* in the signal that contains a transition in the middle. If the transition goes from high to low (falling edge), the bit is a '0'. Similarly, if the transition goes from low to high (rising edge), the bit is a '1'. Differential Manchester encoding considers the same window, while again, there is always a transition in the middle. However, the bit value is determined if there is a transition between the end of one window and the beginning of the next. If there is a transition, the bit is a '0', and if there is no transition the bit is a '1'.

Figure 30 provides an example of Manchester and Differential Manchester encoding of the bit vector "011001." The interesting thing about these two different encodings is that they appear superficially the same, however, unless you know which code is being used, it is impossible to determine the encoding just by examining the data itself. For example, in Figure 30, if a Differential

Table 8: Hardware burst switch detection circuit implemented in 0.16 μm technology at 1.8V.

	2-bit	**4-bit**	**8-bit**	**16-bit**	**32-bit**
Area(#cells)	42	52	78	120	205
Area(μm^2)	124	149	235	327	566
10 MHz Clock					
Power(μW)	14.95	17.82	24.33	33.96	53.45
1 MHz Clock					
Power(μW)	1.78	2.21	2.85	3.84	5.86
100 kHz Clock					
Power(μW)	0.17	0.22	0.28	0.38	0.58
Gated Clock					
Power(μW)	0.11	0.17	0.22	0.32	0.55

Manchester encoding of "011001" is read as Manchester code, it appears to be "101110," which is entirely unrelated to the original value.

To further protect data in our system, we mix the use of Manchester and Differential Manchester encoding in the same data sequence. A key will be used to specify the encoding employed, for example, a '0' would imply Manchester encoding while a '1' would suggest Differential Manchester encoding.

To convert Manchester or Differential Manchester code into bit-vectors, the signal is sampled so as to detect the transition in the middle or beginning of the window, respectively. To extract the values from a signal that combines Manchester and Differential Manchester code in the same signal it is necessary to sample for transitions at both the beginning of the window and the middle of the window, simultaneously. This requires doubling the sampling rate over either encoding alone. Additionally, it will be necessary to create logic to detect the existence or absence of a transition at the beginning of the window and the direction of the transition in the middle of window to determine the value. However, the key can be used as an enable signal to these blocks, so the

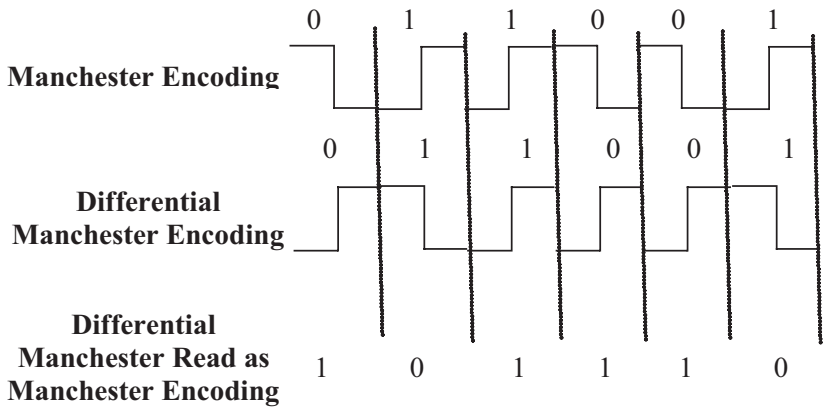


Figure 30: Example of the bit-vector "011001" encoded as Manchester encoding, Differential Manchester encoding, and Differential Manchester encoding read as Manchester encoding.

inactive block does not consume power.

The architecture of the decoder is described in Figure 31. The input signal is sampled to detect the Manchester value and Differential Manchester value for the same window. The Manchester circuit keeps the decoder in synchronization as it detects the transition that occurs in the middle of the window with the direction of this transition dictating the "Manchester" value. It also sends a synchronization signal to the Differential Manchester decoder to tell it when to sample for the beginning of the next window. This is necessary as there is not guaranteed to be a transition between the windows. The existence or lack of a transition dictates the "Differential Manchester" value. The transition detection is completed by some variation of the *edge detection circuit*. This circuit compares the current and previously sampled value and generates a '1' if there is an edge or a '0' if there is not an edge. The direction of the edge can be detected by checking the current or previously sampled value.

The final decoded value is selected by the key, where a '0' means select the "Manchester" value and a '1' means select the "Differential Manchester" value, indicated with the multiplexer in the figure. The bits arrive serially and are grouped into bytes for processing in the tag controller. Keys may be updated by adding new primitives into the RFID system.

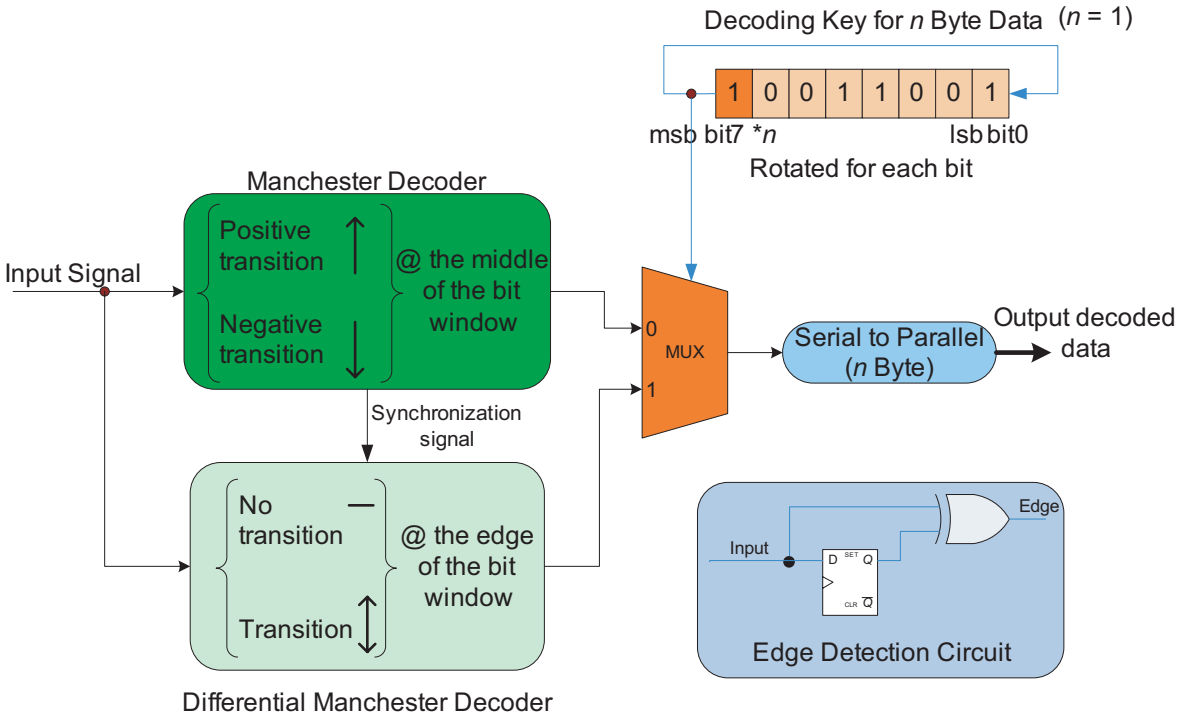


Figure 31: Encoder and decoder block for combining Manchester and Differential Manchester encodings using a key.

4.3.2.1 Results We built a Manchester decoder in 0.16 μm Oki cell-based ASIC hardware synthesized with Synopsys Design Compiler and profiled for power with Synopsys PrimePower. The design was analyzed by simulating the design at 500 kHz. 500 kHz was selected as the appropriate sampling rate for a 27.7 kHz signal, which is the standard data rate for ISO 18000 Part 7 [3]. This design is based on the Manchester decoder developed for a power saving smart buffer for active tags described in [37]. The design was extended as shown in Figure 31 to include concurrent Differential Manchester decoding and selection based on a key. The results of these two designs are summarized in Table 9. The overhead of adding Differential Manchester decoding increases the design area by approximately 42% and the power consumed by 38%.

Table 9: Overhead for adding Differential Manchester decoding using a key to a Manchester decoder.

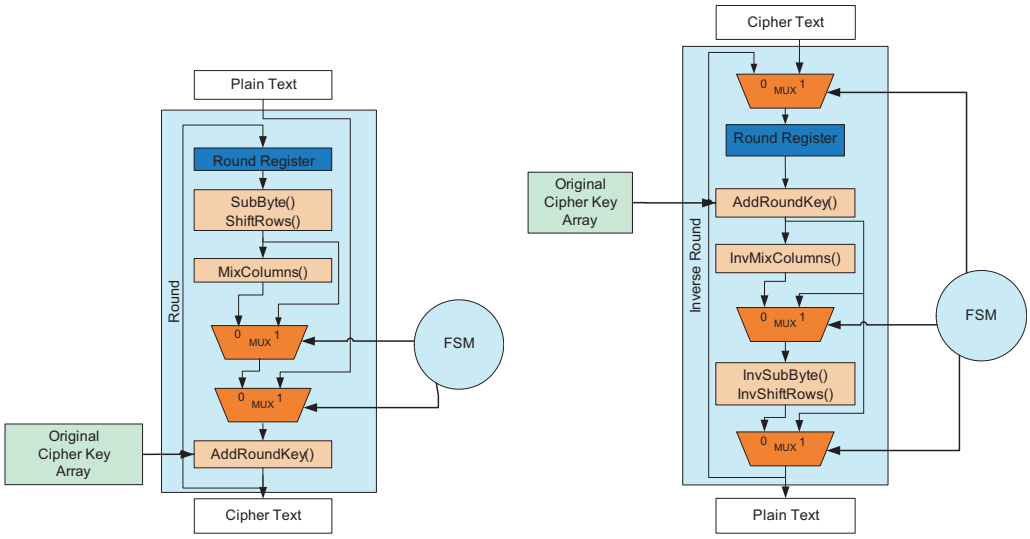
	Area (#cells)	**Area (μm^2)**	**Power (μW)**
Manchester Decoder alone	466	3780	2.886
with Differential Manchester	664	5386	3.996

4.3.3 Encrypted Data Transmission with AES

Once the correct encoding has been received by the burst switch and the physical layer has been successfully decoded creating a packet comprised of several bytes of data, the data must be decrypted. The advanced encryption standard (AES) algorithm from the National Institute of Standards and Technology (NIST) was selected for our third layer of security. Our implementation goal for AES was to create an implementation that was low-energy to allow the longest battery life possible for the tag. As a result, we employed the SuperCISC compiler [89] to assist with the hardware generation for the encryption and decryption completed in the tag.

The SuperCISC compiler generates extremely energy efficient hardware descriptions from C applications. The primary concern for active tags is battery life. It is often inconvenient or impossible to change a battery. Additionally, active tags are typically much more expensive than passive tags, costing on the order of $100, compared to several cents for passive tags. Silicon area in the tag controller design impacts the per part cost of the chip that goes into the tag. The SuperCISC approach often trades area increases for energy savings. However, with the cost of active tags being so high, a slight increase in tag cost versus a longer battery life is easily acceptable.

An overview of the AES implementation approach is shown in Figure 32. A total of five blocks for the C implementation of AES were synthesized using SuperCISC. For encryption, AES requires 12 rounds of computation with four functions occurring back to back: `SubByte()`, `ShiftRows()`, `MixColumns()`, and `AddRoundKey()` with `AddRoundKey()` being executing prior to these rounds and all functions with the exception of `MixColumns()` being executed once after the rounds. Thus for encryption, three blocks were synthesized:



(a) Encryption. (b) Decryption.

Figure 32: AES architectural overview.

`AddRoundKey()`, which is primarily a Galois Field Adder, the combination of `SubByte()` and `ShiftRows()`, which simplify to table lookups and routing, and `MixColumns()`. Using a simple finite state machine (FSM), the resulting encryption block is shown in Figure 32(a). Decryption requires complementary functions `InvMixColumns()` and the combination of `InvSubByte()` and `InvShiftRows()` in addition to the `AddRoundKeys()` block from encryption. Decryption is implemented in a similar manner to encryption, as shown in Figure 32(b).

Figure 33 shows core synthesized hardware for the `MixColumns()` block. For ease of visualization, only a single iteration is shown here. `MixColumns` and its decryption complement `InvMixColumns()` are the most complex blocks in the AES algorithm. Even this function is built from relatively simple computations, constant modulus, constant increment, compare against zero and select, table lookups, and exclusive OR. The SuperCISC compiler provides a low latency combinational implementation of all these blocks to avoid the power overheads of a highly pipelined implementation with a possible increase in area. However, the performance of these blocks far exceeds the relatively slow communication speeds of approximately 30 kHz employed by active tags.

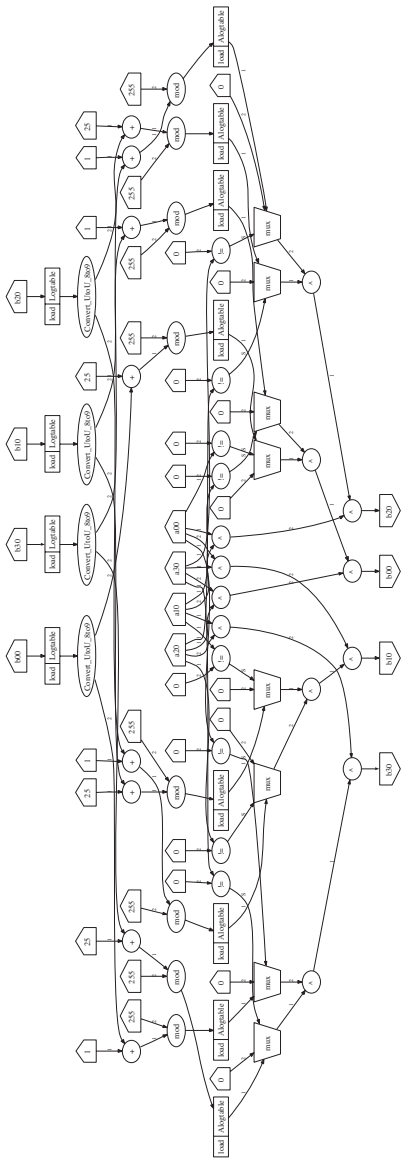


Figure 33: One loop iteration of the `MixColumns()` function from AES encrypt.

4.3.4 Physical Security

RFID tags are often deployed in environments that do not provide physical security for the device. This opens the RFID tag to a class of attacks that involve analysis of the tag power consumption to determine the secret key stored in the device. These types of power attacks have been effective for breaking the encryption of smart cards [57]. The first class of attack is simple power analysis

(SPA), which involves correlating the power consumption of the device to the input provided [46]. The second class of attack is differential power analysis (DPA), which uses statistical analysis to help discover the secret key [47]. While both SPA and DPA require knowledge of the encryption algorithm's behavior, SPA is only effective when the algorithm has a control flow that depends on the secret key, where DPA is effective independent of whether the control flow depends on the secret key [54] making it a much more powerful attack. As a result, we consider the effectiveness of DPA attacks on the RFID tag and discuss methods to protect against these attacks.

4.3.4.1 Differential Power Analysis (DPA) DPA attacks attempt to discover the secret key by subdividing the key into subgroups of a limited number of bits and examining power traces for all the combinations of each of these subgroups. For example, in 128-bit AES, the 128-bit key is subdivided into 32 groups of four bits each. During DPA, a subgroup is examined between "0000" and "1111" with all the remaining groups set to a fixed value. The power analysis compares a known implementation that is using the guessed key value with the system to be broken where the internal value of the key is unknown. DPA takes the difference between these two implementations (averaged over the course of many input text samples) for each guessed key value where on average the power differences cancel out, however, there are spikes when key values match between the two systems.

A more rigorous explanation is based on the descriptions in [47, 57]. The AES algorithm is executed with N values of plain-text input for each guessed key. A discretized power output signal $S_i[j]$, is recorded for each of the guessed keys, where i is a particular plain-text input and j is a particular key. For AES, $0 \leq i < N$ with N typically being 512 and $0 \leq j < 32$ where the upper bound is the product of the number of discrete keys per group and the number of groups. Using the partitioning function $D(\cdot)$, the power traces are subdivided into sets where the guessed key contains a '0' or '1' as follows:

$$S_0 = \{S_i[j] | D(\cdot) = 0\}$$
$$S_1 = \{S_i[j] | D(\cdot) = 1\} \tag{4.3}$$

$$A_0[j] = \frac{1}{|S_0|} \sum_{S_i[j] \in S_0} S_i[j]$$
$$A_1[j] = \frac{1}{|S_1|} \sum_{S_i[j] \in S_1} S_i[j] \tag{4.4}$$

$$T[j] = A_0[j] - A_1[j] \tag{4.5}$$

The average power for each guessed '0' and '1' is computed as $A_i[j]$ as shown in Eq. (4.4). The DPA bias signal, $T[j]$, is the difference of the average power as shown in Eq. (4.5). $T[j]$ is the value which shows spikes when the correct secret key is guessed.

Table 10: Power, energy, throughput, and area of the AES hardware block implemented in Oki 0.16 μm cell-based ASIC design.

AES Mode	**Clock Speed**	**Power**	**Energy**	**Rate**
Encryption:	Area 0.46 mm^2			
	10 MHz	4.62 mW	5.08 nJ	1 Gb/s
	500 kHz	0.19 mW	4.18 nJ	5 Mb/s
	100 kHz	0.047 mW	5.17 nJ	1 Mb/s
	10 kHz	4.056 μW	4.46 nJ	106 Kb/s
Decryption:	Area 0.72 mm^2			
	10 MHz	7.05 mW	7.76 nJ	1 Gb/s
	500 kHz	0.28 mW	6.16 nJ	5 Mb/s
	100 kHz	0.069 mW	7.59 nJ	1 Mb/s
	10 kHz	7.031 μW	7.73 nJ	106 Kb/s

4.3.4.2 Results The AES encryption and decryption components were implemented in 0.16 μm Oki cell-based hardware synthesized using Synopsys Design Compiler and simulated at different speeds with Mentor Graphics Modelsim. The power consumption was analyzed using Synopsys PrimePower. The results are reported in Table 10. The results show that while there is a large variation in power consumption dependent on the speed the energy consumption for the entire AES operation is relatively similar varying between 4.18 and 5.17 nJ for encryption and 6.16 and 7.76 nJ for decryption. If the minimal energy solution was selected, it would be the 500 kHz speed allowing a throughput of 5 Mb/s, which matches the clock speed for the Manchester decoder. However, for line speeds of 27.7 kHz as is defined by the ISO 18000 Part 7 standard for active tags, the 10 kHz clock yielding a throughput of 106 Kb/s is more than sufficient.

This design may also be possible or passive tags, which operate on a very strict power budget. Typically, passive tags consume a few μW of power. The AES design at 10 kHz is already in the μW range, and if the speed is reduced to approximately 30 kHz, the power may be reduced by as much as 3.5-4 times to 1-2 μW.

4.3.4.3 Resistance to the DPA attack It has been shown that the predominant power consuming component of the AES algorithm is the Galois Field Adder, GF(2^n), which is typically implemented as an XOR function. The GF(2^n) adder is from the `AddRoundKey()` function, and the SDFG for this function is shown in Figure 34 as parallel XOR gates. To ensure that our RFID tag is not susceptible to DPA, it is necessary to mask the power signature of this component so that it does not change in the predicted manner dependent on the key.

The standard technique to accomplish the power signature obfuscation is to calculate each bit of the GF(2^n) (XOR) calculation as if the bit is a '0' and a '1' and then use the actual key value to select which result is actually used in the computation. While, this method obfuscates the power consumption, it does require twice as much work increasing the area and power consumption of the device. Our goal is to protect the implementation from DPA while minimizing the area and power increase.

The DPA technique presented by Kocher in [46, 47] assumes that the encryption algorithm is run on a microprocessor and that based on the output testing, the test can determine when the application is processing the portion of the key containing the four-bit window currently being

tested. By using the SuperCISC technique for our implementation, the entire key is operated on in parallel making it impossible to distinguish the operations on the four-bit window from the other 124 bits of the key.

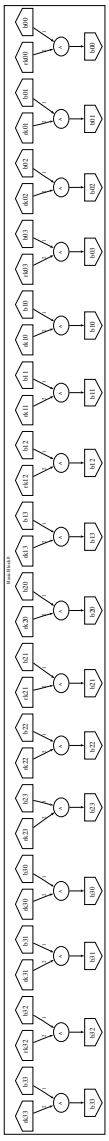


Figure 34: `AddRoundKey()` SDFG, which is basically a 2^n Galois Field Adder.

The DPA attack was applied to our AES design holding the secret key 0x2B7E151628AED2A6ABF7158809CF4F3C. Table 11 shows the DPA analysis (power difference) between our design and a reference design. In the table the rows represent the differences for a particular 4-bit window of the key. The bold values represent the maximum power difference with a particular

key and the asterisk number is the actual key. Based on the results, the predicted key would be 0x4B33D3CD2112179A628B3A11672FB053, which is different than the actual key. To prove that there is no relationship between the actual key and the predicted key we performed a t-Test between the maximum difference magnitude and the difference magnitude when the key matched the secret key.

For our test, we used an alpha value of 0.05 and the resulting p-value was < 0.01, which shows that there is sufficient resolution for the maximum difference and that the maximum difference is unrelated to the difference from guessing the correct key. We also calculated the number of bits that differ between the key with the maximum difference and the correct key. The average number of bits that differ is 2.00 with a standard deviation of 0.95. The median is 2. This shows that even considering the deviation the result is a positive number of bits typically ranging from 1 to 3 bits that differ between the correct and maximum difference guess. This confirms that the guess from DPA of our AES design is truly unrelated to the actual correct key.

Table 11: Differences for each 4-bit window of the 128-bit key. **Maximum difference is in bold** and correct key is denoted with a *.

	Difference in μW for the guessed key															
Window	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	-1	6	5*	1	**12**	5	1	6	1	0	-7	2	-1	-4	-1	-3
1	5	4	6	1	-4	0	-12	-100	3	-1	5	**-108***	1	3	6	-1
2	7	8	5	**-9**	-2	-1	0	-4	2	3	4	0	-4	3	1*	-4
3	7	8*	5	**-9**	-2	-1	0	-4	2	3	4	0	-4	3	1	-4
4	1	3*	-1	-4	-2	-1	0	-1	-4	3	4	-6	-4	**9**	2	-1
5	7	8	5	**-9**	-2	-1*	0	-4	2	3	4	0	-4	3	1	-4
6	7	8*	6	-4	-2	-1	-1	0	2	3	4	-8	**9**	6	1	-4
7	-1	4	9	-5	-4	-1	2*	-4	2	3	3	7	-1	**-8**	-7	-5
8	-1	-1	**6***	-3	-2	-1	5	-4	-4	-3	0	3	-4	-5	1	-4
9	7	**8**	5	-1	-2	-1	0	-4	2*	3	4	0	-4	3	1	-4
10	2	**-9**	1	0	4	7	0	3	8	-7	3*	4	-6	4	-1	-2
11	-6	-8	**-9**	-8	-4	-5	7	5	5	7	4	-3	-4	**-9**	-6*	-7
12	7	**8**	**8**	-3	-2	-3	0	-4	2	3	4	0	-4	3*	1	-4
13	7	8	5*	-4	-2	-1	-2	**-9**	-1	2	4	0	3	1	3	-4
14	-2	2	-1	-5	8	-8	-5	-2	0	**-9**	-5*	4	-2	2	0	-5
15	-1	-1	1	0	3	2	6*	-2	0	-6	**-9**	6	7	5	-2	3
16	6	4	2	-1	-2	3	**9**	2	1	-3	-7*	-7	-3	0	2	-1
17	-2	-8	**10**	4	-2	0	5	1	-1	-2	-1	-6*	-1	7	1	-6
18	-1	-1	5	-4	-3	1	0	5	**9**	3	-1	1	-7	-3	-1	1*
19	-2	-4	10	-1	-5	7	2	-9*	2	4	-1	**12**	-4	1	-5	-7
20	-9	5*	6	**11**	0	-2	7	-2	-4	8	-1	5	-1	-6	3	-2
21	-2	-8	-1	-2	1	-1*	-1	6	2	-5	**10**	6	6	1	**10**	1
22	3	**-11**	9	-4	-2	-6	4	0	2*	1	-8	0	-6	-4	-3	-3
23	3	**8**	-4	-5	0	-2	-1	-4	-3*	0	-3	4	1	2	1	**-8**
24	3*	4	-5	-6	5	3	**11**	3	4	-9	6	8	-8	0	-3	-1
25	-6	2	7	-5	4	-5	-1	**-11**	0	-5*	-5	2	1	0	2	-3
26	1	0	**-7**	-6	4	2	4	-2	-1	1	6	-6	0*	-3	-2	4
27	-1	0	-6	0	3	5	-3	**7**	-4	-1	3	1	-6	2	2	**7***
28	2	3	-4	-2	-5*	1	-3	5	1	1	2	**6**	3	3	2	-2
29	**9**	-4	2	7	-1	0	1	0	4	-5	1	2	-1	7	3	-1*
30	5	2	2	1*	-7	**-8**	0	-1	-2	0	**8**	6	3	-1	1	5
31	-2	-5	-6	**9**	2	-2	1	0	-7	2	-2	5	2*	1	1	4

4.4 CONCLUSION

This chapter describes an overview of the state of the art of security in RFID systems. In particular, the classes of security applied to RFID including various classes of authentication and encryption are outlined in Section 4.2. The types of attacks employed on RFID systems and the facilities

and extensions for securing RFID standards are introduced. In addition, this chapter describes the current security problem for RFID systems, the security and energy trade-off circumstance. As active RFID tags provide More security features, more energy dissipation is consumed. The new security technique is studied and developed as the solution for this problem by providing high degree protection with the minimum power dissipation.

A secure transmission methodology for PART is presented where our primary concern is functionality and power (battery lifetime). Our technique provides layers of security as multiple levels of defense against attack. The passive activation layer is encoded to prevent a malicious reader from continuously activating the tag to drain its battery or to attempt to issue malicious commands. The physical layer is encrypted using a mixture of Manchester and Differential Manchester encoding. Finally, the data itself is encrypted using the AES implemented to avoid the key from being broken using DPA.

These security modifications are implemented with a minimal power overhead from the actual RFID transaction power required. The burst switch detection circuit requires $< 1\mu$W when operating at 100 kHz even for 32-bit values ($.58\mu$W). The modified Manchester, Differential Manchester decoder requires approximately 1μW of additional power (1.11μW). Finally, the AES operation requires a maximum of approximately 7μW at 10 kHz (7.031μW), which is more than sufficient for the data transmission speed. Based on the active tag primitive logic controller implemented in the same hardware, depending on the number of primitives included (up to 40) the power consumed is 63-65 μW [86]. Thus, the overhead for the security is just under 9μW (8.721 μW) which is 13.4% increase over the active tag logic alone.

5.0 FPGA-BASED RFID TRAFFIC PROFILER

5.1 INTRODUCTION

This chapter presents an RFID traffic profile system that records details on the RFID-command information of the RFID communication for the current RFID systems. It allows the users to build the power estimation based on the monitored RFID communication. This technique also provides an affordable approach to profiling RFID traffic, instead of using expensive devices such as digital oscilloscope and network/spetrum analyzer.

To test new RFID systems or implementations, it is valuable to have real workloads from RFID deployments. In order to accomplish this, we have created a technique to generate trace files of RFID communication transactions. These trace files can be used as input test vectors for new RFID components built to satisfy a particular standard. Additionally, these trace files may be adapted for testing new protocols with similar or extended functionality of previous standards.

An overview of the RFID traffic profiling system is shown in Figure 35. As previously mentioned, the RFID interrogator broadcasts RFID commands to the field of tags. The RF-front-end circuit converts the RF signals into the digitized RFID packets. The RFID packets are the inputs to the RFID traffic profile system, which is composed of an FPGA-based RFID-command detector, a C-based command-analysis program and a traffic profiling database.

The RFID traffic profile system is implemented by using as a hardware-software co-design approach. This system consists of an FPGA-based RFID command detector, which is the hardware-portion of the system, and a C-based command-analysis program, which is the software-portion design. The FPGA-based RFID-command detector [90] is capable of decoding the incoming RFID packets. The C-based command-analysis program is composed of an FPGA-related Application Programmer's Interface (API) [91] module and the command-analysis module which is capable

of generating the traffic output file. The C-based software program also allows the RFID network analyst, RFID network manager, or RFID researchers to create a textual output file to study and analyze the RFID communication among readers and tags.

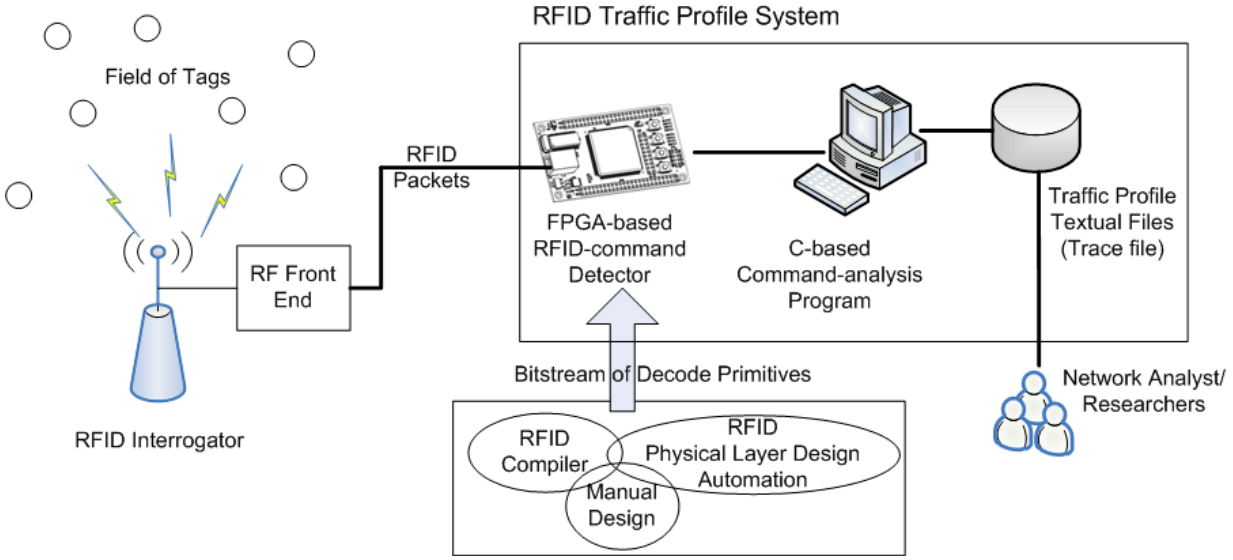


Figure 35: Overview architecture of RFID traffic profile system.

A traffic-profile file or a trace file output from the C-based command-analysis program contains a sequence of RFID commands, also called test vectors, which is captured from real RFID transactions. Therefore, this information can not only be used for network monitoring, traffic analysis, and collision-avoid study, but also can for power estimation of a design.

Multiple RFID standards exist [4, 3, 22, 85, 92] for RFID hardware, software, and data management. Some of these includes active battery powered tags and passive RF energy harvesting tags with frequencies ranging from 125 kHz to 2.4 Ghz. In order to make the FPGA-based RFID-command detector more flexible, the physical layer decoder of the FPGA-based RFID-command detector can be designed and implemented by the RFID compiler [36, 37, 88] and RFID Physical Layer Design Automation tool as described in Chapter 6. They are capable of generating the bitstream file of the VHDL encoder/decoder. Both design automation tools are also capable of reducing design time and cost, making this system appropriate for different RFID standards.

The remainder of this chapter is organized as follows: Section 5.2 contains background material on research and related work pertaining to RFID. The conceptual architecture of the RFID

traffic profiler is presented in Section 5.3 with the description of the hardware-portion and software-portion of the system. The hardware design is covered in Section 5.3.1 with a description of the FPGA-based design approach. The software design of the system is presented in Section 5.3.2 with a description of program's algorithms. Section 5.4 presents an example of using the trace file provided by the RFID traffic profiler to study an insight of RFID transactions. The example describes the power estimation approach for the passive RFID communication described in Section 5.5 including the power estimation results of the RFID communication. The conclusion is organized in Section 5.6.

5.2 BACKGROUND AND RELATED WORK

The targeted RFID system is the EPCglobal Class-1 Generation-2 system [23]. It is a passive RFID system, which uses the UHF radio signal for communications at 860 MHz to 960 MHz.

5.2.1 The EPCglobal Class-1 Generation-2 UHF (Gen2) RFID Standard

Communication from the RFID interrogator (reader) is accomplished by transmitting the *command* to the RFID tag using a standard air interface. The tag will respond by changing the current state and/or transmitting a designated response message to the interrogator.

Figure 36 shows an example of the EPCglobal Gen 2 protocol for inventory and access of a single RFID tag. In step 1, the interrogator (reader) issues a query. In step 2, the tag responds with a randomly generated 16-bit number. This random number is designed to avoid contention between multiple tags and to ensure that the reader is communicating with only a single tag. The reader acknowledges by returning a random 16-bit number in step 3. This selects only one tag with which to communicate. In step 4, only the tag that issued the matching random number responds with its PC/EPC, essentially its identifier. In step 5, the reader issues a transaction request with the same random number. The tag responds with a transaction handle in step 6. In step 7, the actual transaction is issued with the handle as a parameter. Finally, the tag responds to the transaction in step 8.

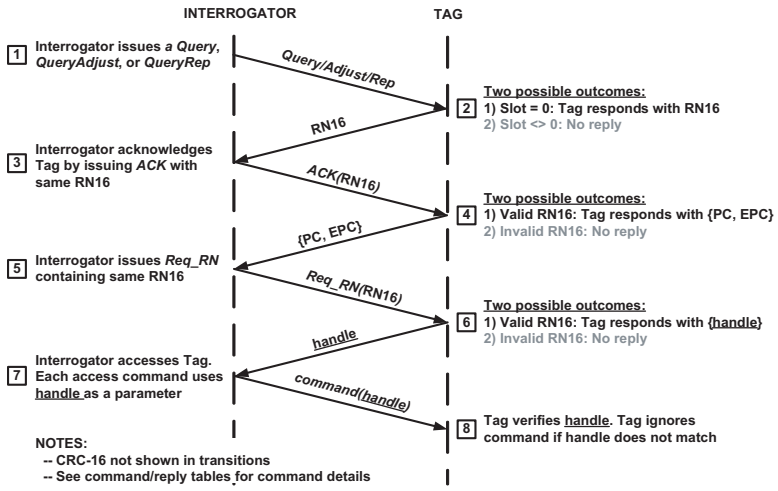


Figure 36: EPCglobal Gen 2 protocol.

An example Gen 2 transaction is shown in Figure 37 [31]. The output shown in this figure is generated by a special piece of equipment housed in the University of Pittsburgh RFID Center of Excellence called a real-time spectrum analyzer (RTSA). The example Gen 2 uses passive RFID technology, requiring the RF energy generated by the reader to be used to power the tag, and to be used for a backscatter-based response. Backscattering uses the reflection of RF energy provided by an external source (in this case the reader) to transmit information. The backscattering device (in this case the tag) either absorbs or reflects the energy of the incoming RF to generate low and high values in the backscattered response.

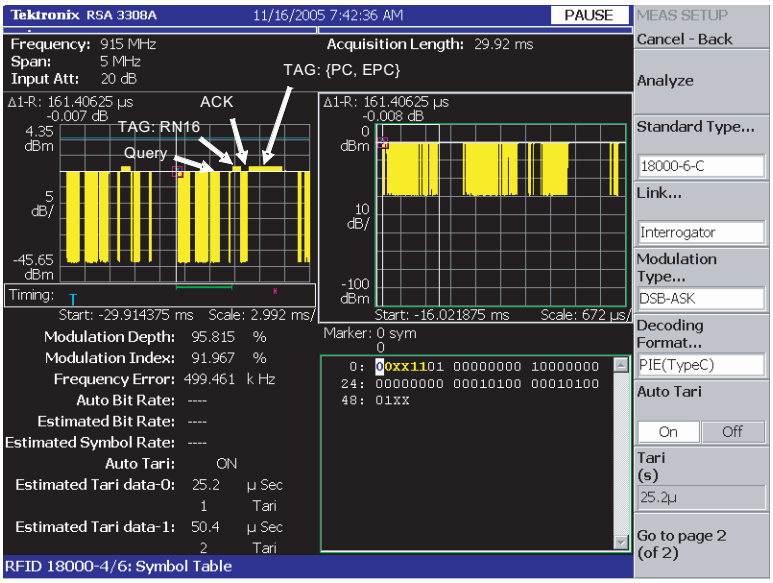


Figure 37: Real-time spectrum analyzer output of a Gen 2 transaction.

5.2.2 Specification of RFID Commands

In order to implement the decoder of the Gen-2 command into the RFID traffic profile system , the fields of the Gen-2 commands are introduced in this section. The communication transactions between the RFID reader and the tag can be broken down into a series of *RFID commands*. The format of the respective fields of each necessary *Gen-2 commands* and its corresponding response are both illustrated in Figure 38.

The command code of each *RFID command* is a unique field or *opcode* that serves as the identifier. Each RFID *command* also contains a subset of fields with varying lengths providing positions for data present in a command in Figure 38. Similarly, the tag response to each *RFID command* has fields of varying lengths.

Query Command

QCmd	DR	M	TRext	Sel	Session	Target	Q	CRC-5
4 bits	1 bit	2 bits	1 bit	2 bits	2 bits	1 bit	4 bits	5 bits

Response

RN16
16 bits

QueryRep Command

QR CMD	Session
2 bits	2 bits

Response

RN
16 bits

QueryAdj Command

QRCMD	Session	UpDn
4 bits	2 bits	3bits

Response

RN
16 bits

Ack Command

ACmd	RN
2 bits	16 bits

Response

PC	EPC	CRC-16
16 bits	96 bits	16 bits

Figure 38: Selected commands and response formats from EPCglobal Gen 2.

5.3 CONCEPTUAL ARCHITECTURE OF RFID TRAFFIC PROFILER

Figure 35 shows the overview of our design flow to generate these trace files. We begin with an commercial off the shelf (COTS) reader, in our case a SAMSys reader [93] from Sirit [94]. This reader allows the raw packets transmitted from the reader and received via the reader from the tags to be connected to an external device. We have connected the reader to a Spartan III FPGA [90] using a wired connection that is detected from by UHF RF portion of the reader. After detecting

the existence of RFID packets, our FPGA-based traffic profile system receives these raw packets and decodes them. These decoded values are then recorded by an external computer into trace files.

The description of the UHF RF detector circuit presented in [95] is placed between the reader and the FPGA board. To translate the UHF signal into a digital signal acceptable for input to the FPGA, a prototype circuit similar to the burst switch circuit described in [31] to produce a digital signal. The circuit is based on the typical design of the demodulator block of a passive RFID device. The detector circuit employs a conventional envelope detector design (a diode-capacitor network) followed by a low-pass filter in order to demodulate the ASK (amplitude-shift keying) signal generated by the reader. Figure 39, captured from the related previous work [95], demonstrates the basic blocks of the circuit design, where the demodulator detects any changes in the voltage amplitude of the input signal. The signal is then passed through the RC filter. For the implementation the time constant of the load components is optimized to minimize the distortion and ensure an accurate demodulation of the data for the selected modulation frequency of the reader. Thus, the digitized RFID packets converted by the RF-front-end circuit (e.g. the UHF RF detector circuit) are the inputs to the RFID traffic profile system.

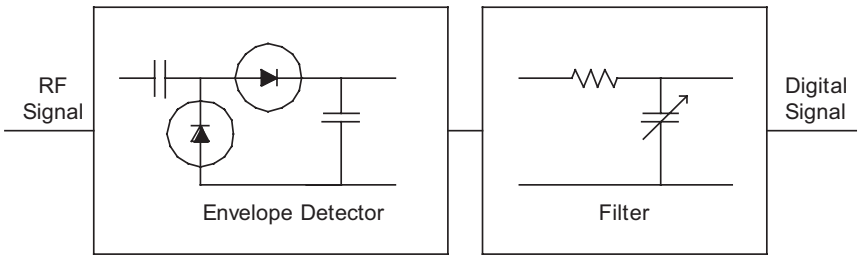


Figure 39: Conversion circuit from UHF signals in the reader into digital signals.

The RFID traffic profile system in Figure 35 is composed of an FPGA-based RFID-command detector, a C-based command-analysis program, and a traffic profile database. The FPGA-based RFID-command detector [90], which is the hardware portion of the design, is capable of decoding the incoming RFID packets. The C-based command-analysis program, the software-portion design, is composed of a FPGA-related Application Programmer's Interface (API) [91] module and

the command-analysis module which is capable of generating the traffic output file. The FPGA device is programmed with a bitstream file which can be generated from three different sources: the RFID compiler, the RFID physical layer design automation flow, and the manual design. The RFID compiler can decode primitives sent from the reader over this wired connection. The RFID physical layer design automation tool can generate the encoder/decoder hardware blocks. The manual design provides the state machine, on-chip memory design, the packet-analysis design, the hardware-software interface design, etc.

Through the physical wire connection and the FPGA-related API module, the C-based software program is capable of performing the burst read of the RFID commands from the on-chip memory bank on the FPGA board. Meanwhile, for each incoming command it also analyzes, decodes and outputs the results into the trace file, which may include the decoded bit information, packet arrive time, etc. The software portion of the design provides flexibility and also allows the RFID network analyst, RFID network manager, or RFID researchers to achieve the textual output file to study and analyze the RFID communication among readers and tags.

A segment of a trace file for testing a single Gen 2 tag with a SAMSys reader is shown in Figure 40. It should be noted that with the circuit from Figure 39, this setup can be performed with any RFID reader. From this trace we can see the behavior of the reader. One interesting result was discovered; the reader actually sends two `select` commands in sequence, which was initially unexpected. However, this is likely the result of the proprietary implementation of the protocol in the SAMSys reader and how it deals with reading large fields of tags. It also demonstrates that the workloads from actual readers can be significantly different even though they conform to the same standard.

The trace file generation was verified using a real-time spectrum analyzer to ensure that the commands issued and stored in the trace file were the same as those broadcast in the air. The remainder of commands shown here follow the expected sequence of the Gen 2 protocol.

Figure 41 presents the conceptual architecture of the hardware-software co-design system. The FPGA-based RFID-command detector is composed of the Preamble Detect Block, the Command Decoder, the Memory Bank Module, the State Machine, and the USB Interface Module. The FPGA device is connected to the personal computer through the USB connection. The C-based command-analysis program, including the FPGA-related API module, the command analysis module, and the

```
Select:1010 000 000 01 00000000 00000000 0 0101000001010001
Select:1010 000 000 01 00000000 00000000 0 0101000001010001
Query:1000 0 00 0 00 00 0 0001 11001
Ack:01 0000000001110010
NAK:11000000
QueryRep:00 00
QueryAdj:1001 00 110
```

Figure 40: Example trace file output from a SAMSys reader and a single tag.

trace generator module, is loaded in the personal computer.

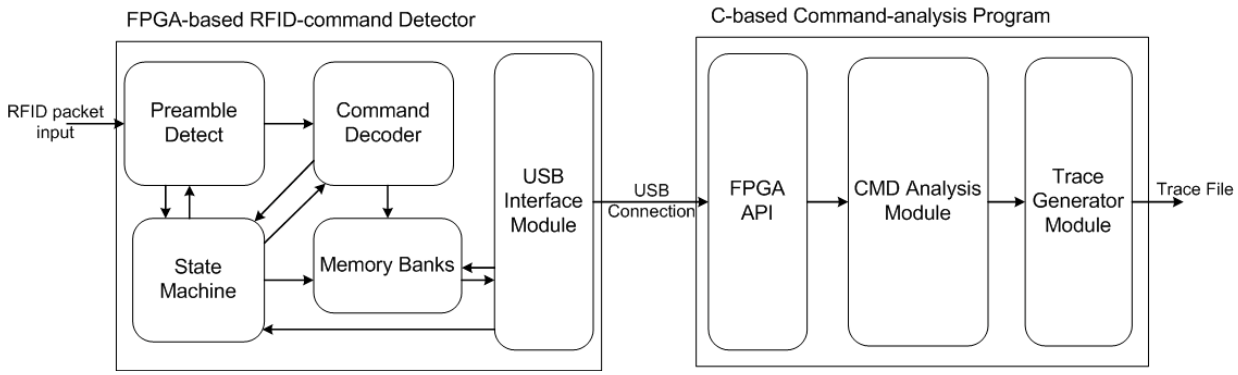


Figure 41: Conceptual Architecture of the Hardware-Software Co-design System.

Since the data flash rate of the FPGA-related API to access the data from the FPGA device is 25ms, it should be noted that the size of the memory bank module on the FPGA chip needs to be sufficient to store the RFID packet information for at least a 30ms period.

5.3.1 Hardware Portion Design

Each component of the hardware-portion design is enumerated as follows:

- **Preamble Detection Block**: detecting the incoming preamble single.

- **Command Decoder**: decoding the RFID command.
- **Memory Bank Module**: storing a sequence of RFID commands.
- **State Machine**: controlling operations in each component, and read/write access to the USB interface module and memory banks.
- **USB Interface Module**: communicating with the C-based command-analysis program.

5.3.1.1 Preamble Detection Block An RFID packet is composed of two segments of bit-stream data. The first segment is a preamble signal and the second segment is an encoded RFID command. The preamble detection block detects a valid preamble signal. It receives digital bit-stream preamble data converted from the RF-front-end circuit shown in Figure 35. This preamble detecting block assures that the RFID traffic profiling system can tolerate the noise and only recognize a valid preamble. A preamble is comprised of a fixed-length unit (**delimiter**) and three non-fixed-length units: a **Tari**, an **RTcal**, and a **TRcal**, as shown in Figure 42 [23]. The specification of the preamble data is defined in the EPC global Class-1 Gen-2 Standard [23]. The RTcal sets the calibration of the transmission from an interrogator to an RFID tag. The TRcal sets the calibration of the backscatter from a tag to an interrogator.

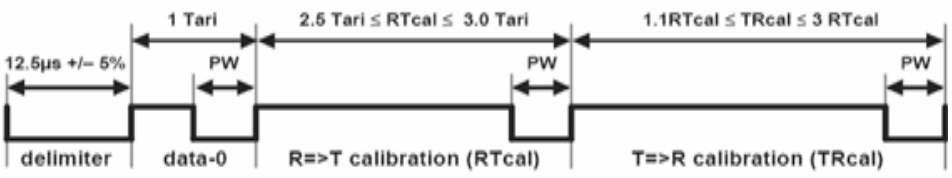


Figure 42: The preamble signal of EPC Global Gen-2 Standard.

The RTcal carries an important information for the RFID traffic profile system. A new value, called a *pivot*, is defined as $pivot = RTcal/2$. The *pivot* is used for decoding the following RFID command. After examining the length of each unit in the bit-stream preamble data, the preamble detect block signals the state machine, computes the *pivot*, and passes this value and the encoded RFID command to the command decoder.

5.3.1.2 Command Decoder After receiving the *pivot* and the encoded RFID command, the command decoder uses the *pivot* to examine the value of each bit symbol. The data encoding used in this ECP standard is the Pulse-Interval Encoding (PIE), as shown in Figure 43. If the length of a bit symbol is shorter than the length of the *pivot*, the bit symbol will be decoded as bit '0'. On the other hand, if the length of a bit symbol is longer than the length of the *pivot*, this bit symbol is declared as bit '1'.

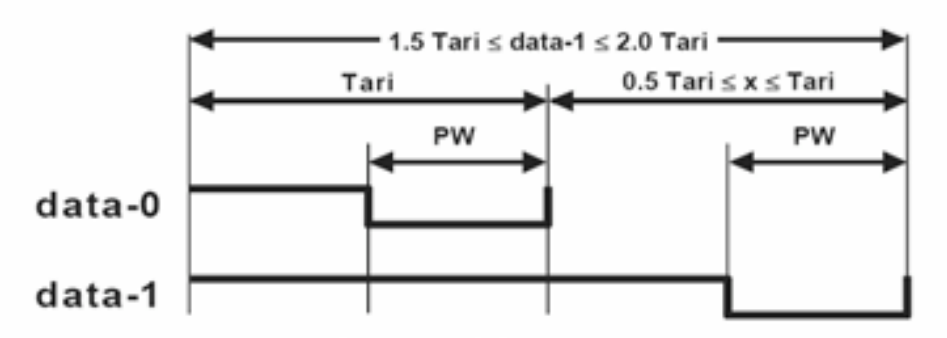


Figure 43: The Pulse-Interval Encoding of EPC Global Gen-2 Standard.

In the EPC global standard, some commands are fixed-length, but some are variable. Each varied-length command contains the **Length** data field in the command. Therefore, the command decoder is not only designed for decoding every bit symbol, but also for recognizing the type of the command and its length in real time.

5.3.1.3 Memory Bank Module The memory bank is a dual-port memory block. The state machine can write the entire command into the memory from port A. Meanwhile, the USB interface controller is able to perform a burst read of the commands from port B of the memory bank.

When the command decoder finishes decoding a command, the entire RFID command will be written into the memory bank. Since the data flash rate of the FPGA-related API to access the data from the FPGA device is $25ms$, it should be noted that the size of the memory bank module on the FPGA chip needs to be sufficient to store the RFID packet information at least for $30ms$ period.

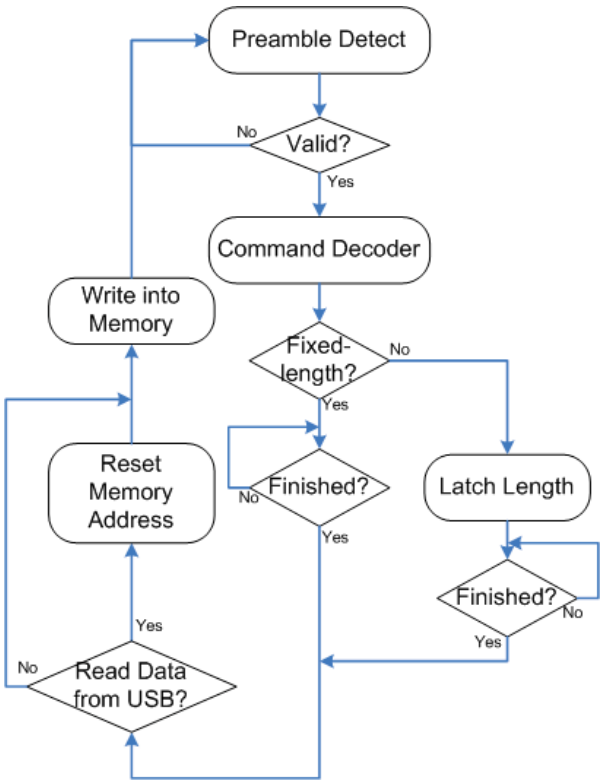


Figure 44: The control flow diagram of the state machine.

The length of the `Lock` command is 60 bits. It is the longest fixed-length command. In order to provide sufficient memory space for storing a 60-bit `Lock` command for a 30ms period, the size of the memory bank should be more than 4800 bits. Therefore, the size of the memory bank is 80X64.

5.3.1.4 State Machine Figure 44 shows the control flow of the state machine. It drops the incoming packet, if the preamble detection block detects that the signal is invalid. For the decoding process, it signals the command decoder to decode bit-by-bit. For decoding the variable-length commands, it sets a length counter when it latches the length of the command. For decoding the fixed-length commands, it decreases the length counter which has been set in the initial process.

Before the state machine writes the decoded command into the memory bank, it checks if any access operation comes from the USB interface. If the memory has been accessed, it resets the

memory address and writes the command into the memory. When the state machine processes an entire command and writes it into the memory, the command counter is increased by one. This counter tracks the number of commands which have been decoded and stored in the memory.

```
-- Opel Kelly Host Interface Component
component okHostInterface port (
        hi_clk          : in    std_logic;
        hi_rdwr         : in    std_logic;
        hi_cs           : in    std_logic;
        hi_irq          : out   std_logic;
        hi_busy         : out   std_logic;
        hi_addr         : in    std_logic_vector(3 downto 0);
        hi_data         : inout std_logic_vector(15 downto 0);
        ti_clk          : out   std_logic;
        ti_control  : out   std_logic_vector(12 downto 0);
        ti_data         : inout std_logic_vector(15 downto 0));
end component;

component okWireIn port (
        ti_clk          : in    std_logic;
        ti_control  : in    std_logic_vector(12 downto 0);
        ti_data         : in    std_logic_vector(15 downto 0);
        ep_addr         : in    std_logic_vector(7 downto 0);
        ep_dataout : out   std_logic_vector(15 downto 0));
end component;

component okWireOut port (
        ti_clk          : in    std_logic;
        ti_control  : in    std_logic_vector(12 downto 0);
        ti_data         : out   std_logic_vector(15 downto 0);
        ep_addr         : in    std_logic_vector(7 downto 0);
        ep_datain   : in    std_logic_vector(15 downto 0));
end component;
```

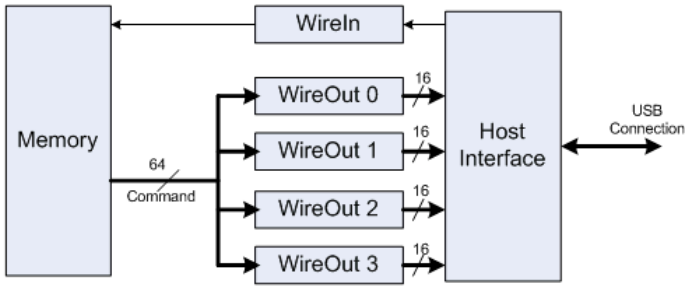


Figure 45: The VHDL components of the USB interface module.

5.3.1.5 USB Interface Module Figure 45 presents the VHDL components for the USB interface module, including the Host Interface, WireIn, and WireOut modules. The Host Interface module uses read/write, chip enable, IRQ, busy, data, and addr signals to communicate with the FPGA-related API program on the software side. It uses clk, data, and control signals to access data from the WireOut and WireIn modules. The WireOut and WireIn modules are I/O buffers. The size of the WireOut module is 16-bit wide. Therefore, 64-bit output is split into 4 segments of data into WireOut 0, 1, 2, and 3. The Host Interface sequentially updates (e.g. read or write) data from the WireIn and WireOut modules.

5.3.2 Software Portion Design

The hardware portion of FPGA-based RFID traffic profiler design is responsible for capturing the incoming commands, decoding them, and storing them into the memory banks. The software portion design is responsible for reading the incoming commands from the memory, analyzing them,

and generating the trace files for these commands. The software portion design is consisted of *FPGA-related API*, *Command Analysis Module*, and *Trace File Generator Module*. Each component of the software-portion of the design is enumerated as follows:

- **FPGA-related API**: communicating with the FPGA-based RFID-command detector.
- **Command Analysis Module**: analyzing each RFID command accessed from the FPGA device.
- **Trace File Generator Module**: generating the trace files.

5.3.2.1 FPGA-related API Module The FPGA-related API module is capable of exchanging data with the FPGA device through the USB connection. Figure 46 shows the flow of building the connection to the FPGA device. If the FPGA-related API module receives the correct information such as `dll_date`, `dll_time`, and the PLL clock frequency, then the API module has established the connection to the FPGA device. The command analysis module is then ready to access the memory on the FPGA chip.

```
// Load FPGA library
okFrontPanelDLL_LoadLib(NULL);

//Test the connection
okFrontPanelDLL_GetVersion(dll_date, dll_time);

//Initialize the FPGA device
okCUsbFrontPanel *xem;
xem = initializeFPGA();

//Set the PLL file and achieve it from the FPGA device
xem->SetEepromPLL22150Configuration(pll_default);
xem->GetEepromPLL22150Configuration(pll);
```

Figure 46: Configuration of building the connection to the FPGA device.

5.3.2.2 Command Analysis Module Figure 47 shows the algorithm to read the commands from the memory, analyze each of the commands, and output them into the trace file. First, while any command exists in the memory bank, the command analysis module gets the command from the memory bank on the FPGA device. The `xem` is declared as the object of the FPGA device. The

GetWireOutValue, a member function of the xem, reads the data from the WireOut module, returns the integer value of the data, and stores the integer value of the data in the variable, the CMD_value.

```
while (NO_of_CMD != 0)
{
  int CMD_value = xem->GetWireOutValue;
  char* CMD_str = Convert_int_to_ASCII(CMD_value);
  Analyze(CMD_str)
  {
    Decode CMD_code;
    Format CMD_str;
  }
  Output(Trace file);
}
```

Figure 47: Configuration of building the connection to the FPGA device.

The convert_int_to_ASCII function convert the int value of the command into a command string, which is a character array of ASCII characters, '1' and '0'. The CMD_str is a character string variable storing the returned character array from the convert_int_to_ASCII function.

The analyze function decodes the command string. Based on the different command codes, the analyze function recognizes each command string and presents each of them in its particular format. The example in Figure 40 shows multiple decoded commands organized in their particular formats. For instance, the Query command is formatted as CMD_code DR M TRext Sel Session Target Q CRC5 and presented as 1000 0 00 0 00 00 0 0001 11001.

5.3.2.3 Trace File Generator Module The output function generates the trace file. It can be used not only to print the decoded commands sequentially, but also to output groups of commands into the trace file. For example, the user can specify that the output function groups the commands by the command code or lists them by arriving time.

5.4 CASE STUDY 1: Q VALUE VERSUS SLOT COUNTER

An RFID interrogator uses `Query` to initiate an inventory round. Figure 36 shows an inventory round beginning with a `Query` command sent by an interrogator to a tag. A `Query` command carries a Q-value within the command, which the reader can set. Upon receiving a `Query`, the tag generates a random value in the range of (0, $2^Q - 1$), and loads this value into its slot counter. If a tag loads the slot value with zero, it replies an RN16 number to an interrogator. (An RN16 number is a 16-bit random number.) Otherwise, the tag shall remain silent until its slot counter is decremented to zero.

For example, if an interrogator sends `Query` with Q = 1, a tag is able to pick up the number from the range of (0, 1). Thus, the slot counter is either 0 or 1. If Q = 3, the tag is able to pick up the slot counter from the set of (0, 1, 2, 3, 4, 5, 6, 7).

The `QueryRep` command instructs tags to decrement their slot counters. If the slot counter equals zero after decrementing, the tag backscatters an RN16 to the interrogator.

Figure 48 and Figure 49 present two segments of a trace file illustrating the initiation of two different inventory rounds. An example in Figure 48, an RFID reader initiates an inventory round beginning with the `Query` command with Q = 1. After sending a `QueryRep` command to decrement the slot counter, the reader receives an RN16 and sends the `Ack` command carrying this number back to the tag. Therefore, the slot counter in the tag is 1.

```
Select:1010 000 000 01 00000000 00000000 0 0101000001010001
Select:1010 000 000 01 00000000 00000000 0 0101000001010001
Query:1000 0 00 0 00 00 0 0001 11001
QueryRep:00 00
Ack:01 0000000001110010
```

Figure 48: A trace file including Query and QueryRep when Q = 1.

In Figure 49, an RFID reader initiates an inventory round beginning with the `Query` command with Q = 3. After sending five `QueryRep` commands to decrement the slot counter, the reader receives an RN16 and sends the `Ack` command with this number back to the tag. Therefore, the

```
Select:1010 000 000 01 00000000 00000000 0 0101000001010001
Select:1010 000 000 01 00000000 00000000 0 0101000001010001
Query:1000 0 00 0 00 00 0 0011 01011
QueryRep:00 00
QueryRep:00 00
QueryRep:00 00
QueryRep:00 00
QueryRep:00 00
Ack:01 0110010001110010
```

Figure 49: A trace file including Query and QueryRep when Q = 3.

slot counter in the tag is five.

Through this case study, the RFID traffic profile system is capable of providing internal information for each RFID communication. Although this case study is targeting the passive RFID system, the RFID traffic profile system can also be used in active RFID systems.

5.5 CASE STUDY 2: POWER ESTIMATION OF RFID COMMUNICATION

Many communication parameters such as the size of the command, the complexity of the communication, the number of retransmissions, etc, can have an impact on the power consumed in each tag. The power consumption of a RFID tag can be effected significantly by the behavior of an RFID reader with respect to the commands sent by the reader. The RFID traffic profile system can be used to evaluate such power consumption.

In this case study, the FPGA-based traffic profiling system is used to monitor the RFID communication transactions for a two-second period and outputs the transactions into a trace file. The first row in Table 12 shows the number of commands found in the trace file.

Table 12: Power consumption of RFID commands and energy estimation for RFID transactions.

RFID Commands	**Query**	**QueryAdj**	**QueryRep**	**Nak**	**Ack**
# of CMDs	89	71	261	14	36
Traditional Method					
Power (mW)	0.113	0.059	0.061	0.034	0.052
Length (μs)	275	112.5	50	100	225
Energy(nJ)	31.075	6.6375	3.05	3.4	11.7
Total Energy(μJ)	2.765	0.471	0.796	0.047	0.421

Table 12 also shows the power consumption of each command from the EPC Global Gen-2 standard. Upon the number of commands and the power consumption number, the energy estimation for RFID communication transactions from the trace file can be calculated in the row of **Total Energy**. Each of the RFID commands is synthesized by using Synopsys Design Compiler. The power profile of each RFID command is generated by simulating each design in Mentor Graphics ModelSim and estimating the power with Synopsys' PrimePower.

From the EPC Global Gen-2 standard, the length of each command can be calculated and listed in the fourth row in Table 12. The data for the entire RFID communication is captured within two seconds by running the "auto read" command in the SAMSys Reader, as shown in Figure 35. The number of each command sent by the reader can be calculated automatically by the C-based Command-analysis program. Those numbers are listed in the **# of CMDs** row in Table 12.

For a two-second period, a tag needs to consume 2.765 μJ for processing `Query`, 0.471 μJ for processing `QueryAdj`, 0.796 μJ for processing `QueryRep`, 0.047 μJ for processing `Nak`, and 0.421 μJ for processing `Ack`.

5.6 CONCLUSION

This chapter describes an overview of the RFID traffic profile systems and presents the hardware-software design approach to developing the system. In particular, for the RFID network analysts or RFID researchers, this system provides insight to the communication transactions and outputs

the trace file for captured communication transactions between a reader and tags. The hardware components, such as **Preamble Detector** and **Command Decoder**, can not only be designed by hand, but can also be implemented by the Physical Layer Design Automation tool, described in Chapter 6.

Two case studies illustrate the usage of the trace file generated by the RFID traffic profile system. Based on the trace files, a user is capable of understanding the details on the communication transaction. For example, in order to study the collisions between tags, it is important to understand the information of the slot counter from tags. Every tag has its own methodology to pick the slot counter or to generate the slot counter based on the Q value. However, there is no way to obtain the slot counter value by using a traditional network analyzer or spectrum analyzer. The RFID traffic profile system provides an alternative approach to obtain this information for the users.

The second case illustrates the power estimation for the RFID communication by utilizing the RFID traffic profile system. By sending "auto read" command from the SAMSys reader for two seconds, a tag needs to consume 2.765 μJ for processing `Query`, 0.471 μJ for processing `QueryAdj`, 0.796 μJ for processing `QueryRep`, 0.047 μJ for processing `Nak`, and 0.421 μJ for processing `Ack`.

6.0 RFID PHYSICAL LAYER DESIGN AUTOMATION (PLDA)

6.1 INTRODUCTION

RFID systems have become a ubiquitous technology with applications including logistics, supply chain management, library item tracking, medical implants, road tolling, building access control, aviation security, and homeland security. A wide range of extensions such as memory, sensors, encryption, and access control can be added to RFID tags. The interrogators query the tags for information stored on them, which can include items like identification numbers, user written data, or sensory data.

Multiple RFID standards exist [22, 3, 4, 85, 92] for RFID hardware, software, and data management. Active battery powered tags and passive RF energy harvesting tags are operated with frequencies ranging from 125 kHz to 2.4 GHz. However, most RFID systems are being deployed for closed loop applications and use either proprietary protocols or non-intersecting standards with non-reusable tags and readers. RFID specifications and standards are often vague enough to create compliant tags that cannot inter operate among vendors. Thus, in many applications, RFID tag and reader hardware and software must be specifically designed for each particular application. The entire system must be physically modified or re-designed every time when the specification for the current application is adjusted. In particular, as new applications are introduced, the standards are modified, new standards are developed, and new compliance testing is introduced, RFID system providers need to update or redesign tags or readers. This keeps the overall design time long and the system costs high.

This chapter presents an RFID physical layer design automation system and power estimation system that allows the design, optimization, and verification of new RFID specifications and standards. This technique also allows the merging of two or more standards or specifications for

interoperability among standards and backward compatibility. The resulting RFID device consists of an automatically generated controller and physical layer hardware block that can interface with existing RF circuitry and antennas.

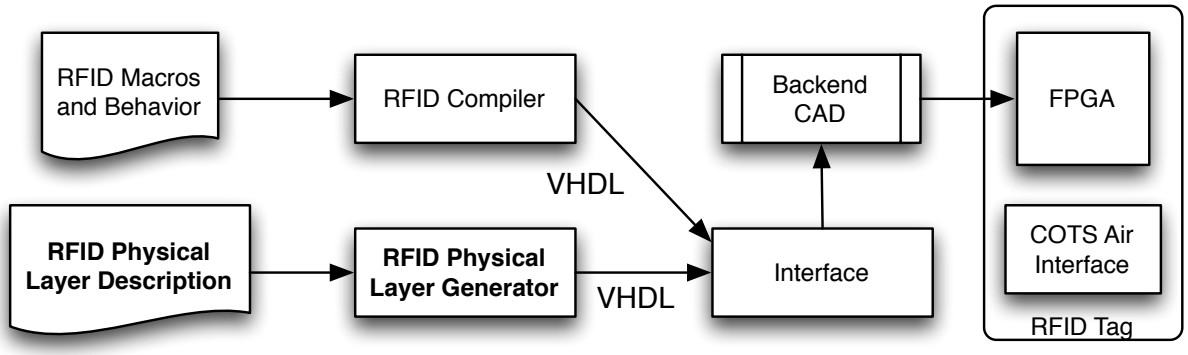


Figure 50: Overview of the RFID design automation flow.

The physical layer design automation tool also integrates with a design automation flow for the transaction level of the RFID system [36, 37, 31, 86], and a commercial off the shelf air interface to create a complete RFID system. An overview of the entire design automation flow is shown in Figure 50 with the physical layer design flow highlighted.

Basic RFID systems, either readers or tags, contain three major components: an antenna/air interface, a physical layer encoder and decoder, and a transaction layer controller. In this chapter a design automation flow for the physical layer component of an RFID system is described. This technique is called **Physical Layer Design Automation**. The overview design flow of Physical Layer Design Automation is shown in Figure 51. It indicates several main components used in the tool: the input textual file describing the **physical layer waveform feature**, the **waveform feature library** which provides template VHDL components, and the **synthesis** process of parsing the textual file and generating encoder/decoder VHDL code.

The design methodology allows the users to input the specification of *waveform features* of encodings, which are captured in a textual format. These features specify characteristics of waveforms, such as bit period(s), bit accuracy, edge transitions, level detection, pulse width detection,

etc. They are then used as inputs to parameterized hardware blocks and combined with automatically generated customized hardware blocks. The result is automatically generated encoding and decoding hardware blocks.

The main contributions of the physical layer design automation are to compile a variety of encoding specifications and to generate corresponding encoder/decoder hardware blocks. Each encoding scheme has its own waveform properties. Some of them require one transition in one bit window, but some of them require multiple transitions. Some of them use a fixed bit-window size, but some of them use a floating bit-window size. For some of the encodings, each bit has a particular dependency with the previous bit. But some of them do not follow this rule. Thus, every waveform-feature textual file provided by a user may contain a variety of waveform requirements.

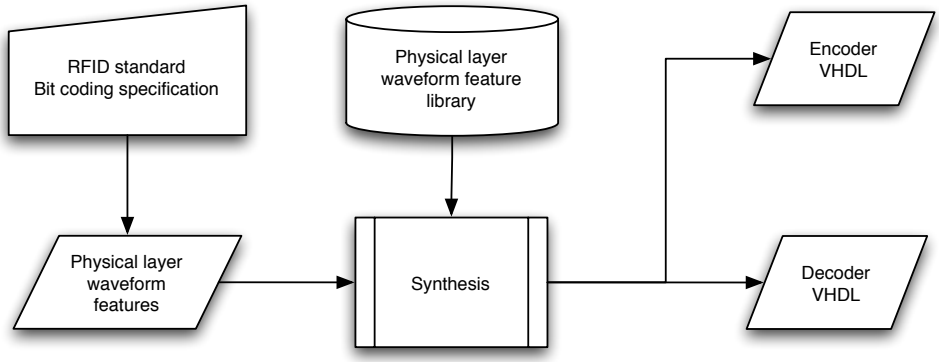


Figure 51: The generation flow of the **Physical Layer Design Automation** system for an RFID data encoder and decoder.

The entire design system was developed using the object-oriented programming language, C++. Two main functions are used in the synthesis process: *Parse* and *VHDL Generation*. The *Parse* procedure is a function used to parse the waveform features and to construct a group of feature objects. The *VHDL Generation* procedure explains how to generate encoder/decoder VHDL code by using the group of feature objects and the template components in the waveform-feature library.

The remainder of this chapter is organized as follows: Section 6.2 contains background material on research and related work. The physical layer design automation flow is presented in Section 6.3. The textual representation of the waveform is described in Section 6.3.1. The parsing

procedure is presented in Section 6.3.2. A description of the interface with the waveform features library and the VHDL generation procedure are described in Section 6.3.3. An explanation of the final synthesis process is presented in Section 6.3.4. Results are presented in Section 6.4. Finally, conclusions are included in Section 6.5.

6.2 BACKGROUND AND RELATED WORK

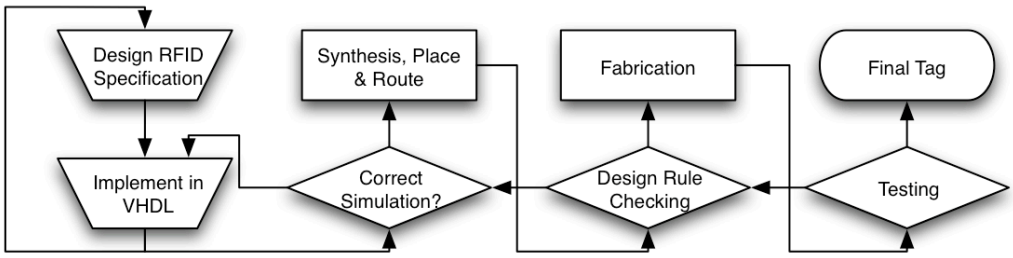
There has been an explosion of interest in RFID in recent years. Some of the open issues in the RFID domain are the existence of multiple standards, the use of high frequency (HF) or ultra high frequency (UHF), the use of near field or far field powering and communication, the handling of stored data and their formats, tag orientation, reader collision [96], range, cost, and security concerns [97]. For active tags, maximizing battery life is an important concern. Recent research has been focused on finding solutions for some of these issues.

Recent advancements in tag hardware contribute to improvements in range, power consumption [31], tag antenna design [98, 99], packaging, and tag orientation. For performance characterization of RFID systems under active interference, a test protocol is presented in [100] and its effectiveness is verified. For RFID readers, a solution is developed for the problem of Tx/Rx isolation at the physical layer [101].

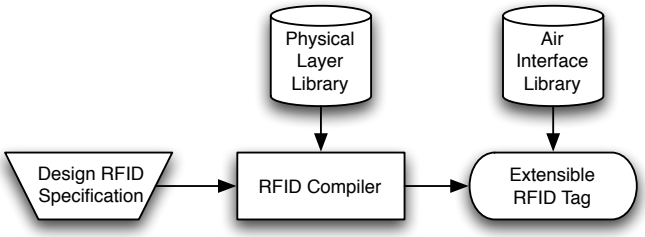
Another issue is that proprietary hardware and software are used in specific application domains. These devices must be physically modified or re-designed for adjustments in the specification, the introduction of new applications, and/or modifications to relevant standards. A customizable RFID tag can handle variations in standards and requirements as they are developed with a significantly shorter time to market than current *ad hoc* techniques such as the prototyping described in [88].

Figure 52 presents a comparison of different RFID tag design methodologies. The current state of the art tag development shown in Figure 52(a) requires lengthy design, fabrication, and testing cycles which can take months, with intellectual property (IP) reuse, to years if developing new IP. A customizable RFID tag, as shown in Figure 52(b), can handle variations in standards and requirements as they are developed with a significantly shorter time to market than current flows.

The RFID compiler in the automated RFID tag design flow is capable of parsing the "Design RFID Specification" into the C language and converting C to VHDL code.. Such a tag is mass produced and tailored to a particular RFID use after fabrication. With the use of automation to program the device, the design time can be reduced to hours or days.



(a) Current RFID tag design flow. All tag components integrated manually. Estimated time: months or years.



(b) Automated RFID tag design flow. Prepackaged extensible silicon device. Estimated time: hours or days.

Figure 52: Comparison of RFID tag design philosophies.

The overview architecture of an extensible RFID tag system is presented in Figure 53. The tag can be easily customized to work with different existing or newly developed standards and even concurrently with proprietary commands tailored to the desired application. The tag consists of automatically generated controller and physical layer hardware that works with existing air interface blocks. To generate the controller and physical layer blocks, a design automation methodology has been developed.

For the controller, the methodology allows the commands employed by the RFID system to be specified using *RFID macros*, an assembly-like format. These RFID macros are processed to generate a template file to specify the behavior for each primitive or macro. All behavior is

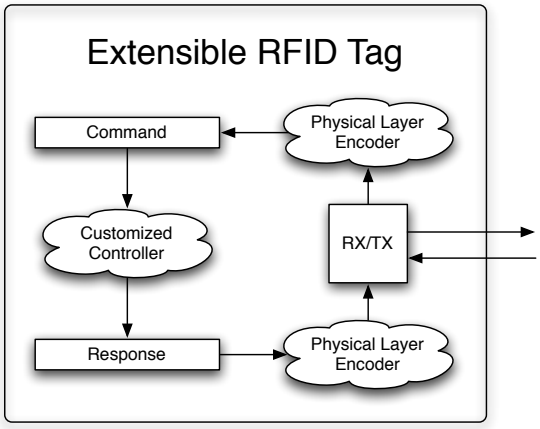


Figure 53: Extensible RFID tag. Balloon objects are automatically generated with RFID Compiler.

specified using ANSI-C allowing the user to create arbitrarily complex behaviors. Finally, the RFID compiler generates the final controller used for managing the tag. This technique allows the seamless coexistence of several RFID standards such as the American National Standards Institute (ANSI) standard 256-2001 [4] and the International Standards Organization (ISO) standard 18000 Part 7 [3]. This technique is described in detail in Dontharaju's dissertation [102].

For the physical layer, the methodology allows the specification of *waveform features* of the encoding, which are specified in a textual format. The waveform feature is defined as a specification of waveforms including signal and timing information. These features are then used as inputs to parameterized hardware blocks and combined with automatically generated customized hardware blocks. The result is automatically generated encoding and decoding hardware blocks.

6.3 RFID PHYSICAL LAYER DESIGN AUTOMATION

One of the main components of an RFID system communication is the physical layer protocol employed to encode bits of information. The physical layer features for the bit encoding mechanism

vary across various RFID standards. For example, the ISO 18000 Part 7 active tag standard specifies *Manchester encoding* [103] to transmit encoded data among RFID interrogators and tags [3], while the ISO 18000 Part 6C standard defines different physical layer features of transactions among readers and tags. *Pulse-Interval Encoding (PIE)* [22] is utilized to encode data transmitted from readers to tags, and either *FM0* [104] or *Miller encoding* [105] is utilized to encode the backscattered data from tags back to readers [22]. Additionally, many other possible physical layer encodings can be considered for RFID communications.

This section describes how the physical layer decoder and encoder blocks can be automatically generated from a high-level specification of the protocol. This design flow is described in Figure 51. For a physical layer specification, the user describes the *waveform features* of the encoding scheme such as edge transitions, level detection, pulse width detection, etc. The user can then combine one or more wave features to represent bits or groups of bits. The physical layer design automation tool, also called as the physical layer synthesis tool, then automatically generates hardware blocks for encoding and decoding the signal in VHDL. These VHDL descriptions are created from the combination of predefined parameterized hardware libraries and automatically generated hardware blocks used for detecting and generating the waveform features in the encoding.

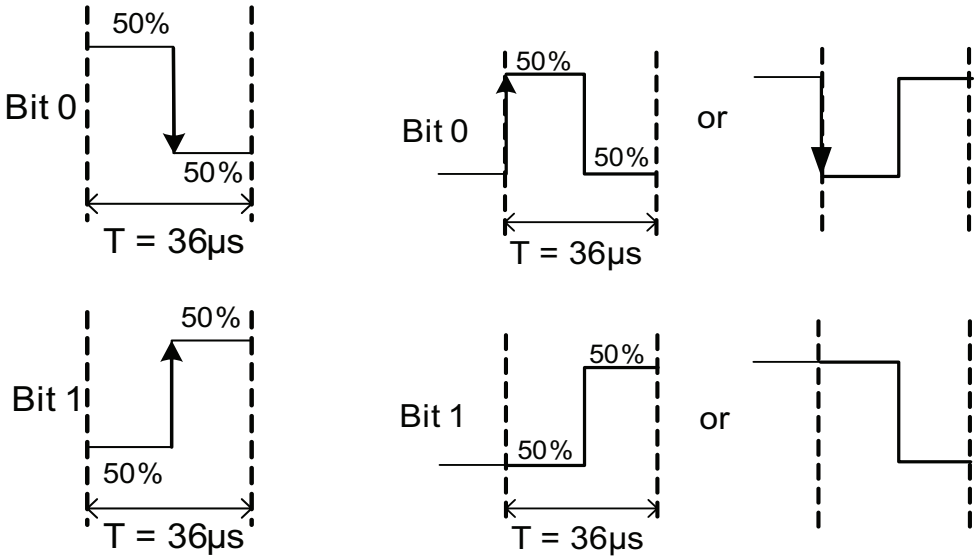
This dissertation presents the design flow for the physical layer from *waveform features* to encoding and decoding hardware blocks. The algorithm, approaches, and techniques used in the design flow are described in this dissertation. The main emphasis of this contribution is to shorten the design time and to provide flexibility to the design flow as well. Thus, the Physical Layer Design Automation (PLDA), not only generates encoding and decoding hardware blocks for different RFID standards, but also for general applications, as long as users provide the specification of *waveform features*.

6.3.1 Specification of Waveform

The user describes the features of the encoding scheme using a textual representation. This representation is created from a physical layer specification such as an RFID standard. It may include edge transitions, level detection, pulse width detection, etc. After this file has been created, it be-

comes the input to the physical layer synthesis tool shown in Figure 51. The textual file contains three major segments: (1) declaration of the waveform to encode data values, (2) the declaration of the preamble waveform, and (3) transmission characteristics for serial to parallel conversion. Five encoding/decoding mechanisms collected from multiple RFID standards are enumerated as following: (1) *Manchester Encoding*, (2) *Differential Manchester Encoding*, (3) *Pulse-Interval Encoding (PIE)*, (4) *FM0*, and (5) *Modified Miller*.

6.3.1.1 Manchester Encoding *Manchester encoding* [103] is a fixed-bit-window encoding scheme specified in ISO 18000 Part 7 for transactions among active RFID readers and tags. The waveforms of encoded bit '0' and bit '1' are illustrated in Figure 54(a). A Manchester encoding signal encodes original data by XORing it with a clock signal. A transition (either a falling or a rising edge) must occur in the middle of each bit window.



(a) Manchester encoding.

(b) Differential Manchester encoding.

Figure 54: Continuous waveform for bits '0' and '1' of NRZ encodings.

The waveform feature of Manchester encoding for encoding a bit as either '0' or '1' is described in Figure 55. `Sig` represents the non-return to zero (NRZ) value of the signal. The keyword `for` describes the period of time for either logic level '1' or '0'. For example, the statement,

`'1' for 18 us`, implies that the signal stays high for 18 μs. Transitions in the signal are represented by an `&` with a non-zero `for` parameter. Thus, the entire feature description,`'0': Sig = '1' for 18 us & '0' for 18 us;`, can be interpreted as the bit '0' is encoded as the signal staying at a high level for 18 μs followed by a falling-edge transition and the signal staying at a low level for 18 μs.

The length of the *bit window* is specified by a period `T`. Finally, `A` specifies how accurately each measurement must be as a percentage of `T`. In the example from Figure 55, a '0' is represented by a 50% duty cycle clock with a falling edge in the middle of the bit window. The edge must be within 12.5% of the total period, which means 6.75% of the period before or after the expected transition. In this case the transition occurs at $18 \pm 2.43\mu$s. A '1' is similar except with a rising edge.

```
'0': Sig = '1' for 18 us & '0' for 18 us;
'0': T = 36 us;  A = 12.5%;

'1': Sig = '0' for 18 us & '1' for 18 us;
'1': T = 36 us; A = 12.5%;
```

Figure 55: Textual description of Manchester Encoding

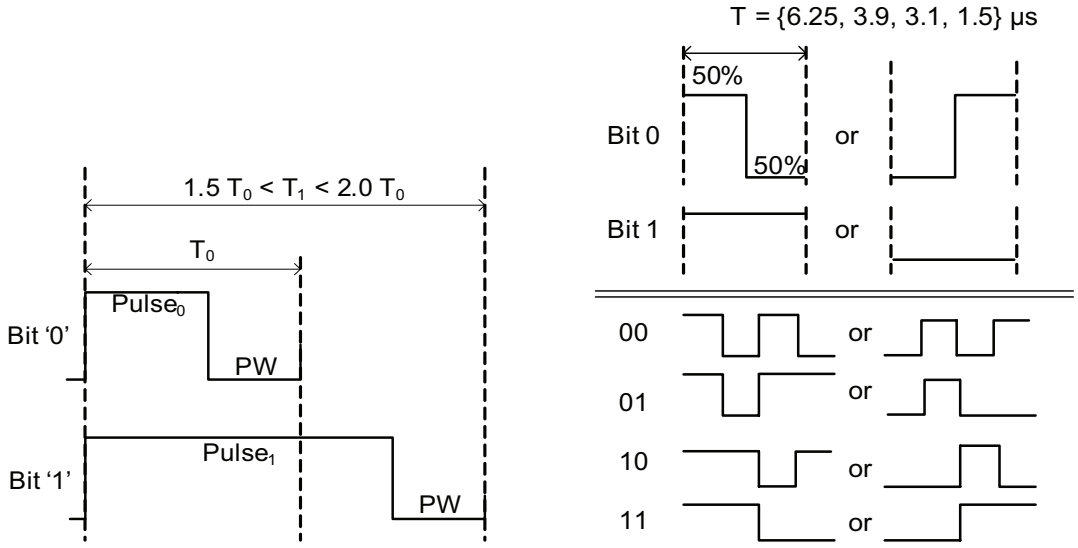
```
'0': if Lprev = '0' then Sig = '1' for 18 us & '0' for 18 us;
     if Lprev = '1' then Sig = '0' for 18 us & '1' for 18 us;
'0': T = 36 us; A = 12.5%;

'1': if Lprev = '0' then Sig = '0' for 18 us & '1' after 18 us;
     if Lprev = '1' then Sig = '1' for 18 us & '0' after 18 us;
'1': T = 36 us; A = 12.5%;
```

Figure 56: Textual description of Differential Manchester encoding.

6.3.1.2 Differential Manchester Encoding *Differential Manchester encoding*, shown in Figure 54(b), is a modification of Manchester encoding used most prominently in token ring networks [106]. At first glance, Manchester and Differential Manchester encodings are indistinguishable as they both require a transition in the middle of a bit window for synchronization, and may or may not have a transition at the edge of a window. However, Differential Manchester determines its value by examining the level between windows. As shown in Figure 54(b), if there is an transition between windows this encodes a '0' and if there is no transition this encodes a '1'.

To support this case, we add an `Lprev` condition to our waveform representation as shown in Figure 56. Depending on the previous level, the user's description describes a level change for a '0' encoded bit and no level change for a '1' encoded bit. The `if` statement determines which waveform to consider based on `Lprev`.



(a) Pulse-Interval encoding (PIE) encoding.

(b) FM0 encoding.

Figure 57: Continuous waveform for bits '0' and '1' of PIE and FM0 encodings.

6.3.1.3 Pulse-Interval Encoding (PIE) *Pulse-Interval Encoding (PIE)* and *FM0* encodings are two different encodings used in the ISO 18000 Part 6C standard. These encodings are shown in Figure 57. The waveform of *PIE* encoding is shown in Figure 57(a). This encoding is used for the data transmission from an interrogator to RFID tags. The special characteristic of this encoding

is that the bit window of either bit '1' or '0' is not fixed. The physical layer characteristics of PIE are shown in Figure 57(a). Encodings for both '1' and '0' are based on an active high pulse followed by a fixed width space called `PW`. The length of the pulse determines whether a '1' or '0' is encoded. Thus, the period `T` is different for each value. Unlike previous encodings, there is a large amount of flexibility in the pulse lengths to make a valid PIE encoded value.

For PIE encoding, in the textual representation shown in Figure 58 the `error` keyword is introduced, that allows a transition to take place within a range of times. For example, `7.5 us error 1.6 us` means that the transition could occur anywhere from 5.9 μs to 9.1 μs. This is different from the `A` which corresponds to jitter associated with the RF transmission. For the PIE encoding described in the ISO 18000 Part 6C standard, there are three possible periods for encoding the values. Each is described in a separate statement in Figure 58. In some cases, the period itself may fall within a range which is described by the `error` keyword.

[Tari = 6.25μs]

```
'0': Sig = '1' for 3.125 us & '0' for 3.125 us error 0.64us ;
'0': T = 6.25 us, A = 1%;
```

[Tari = 12.5μs]

```
'0': Sig = '1' for 6.25 us & '0' for 6.25 us error 1.62us ;
'0': T = 12.5 us, A = 1%;
```

[Tari = 25μs]

```
'0': Sig = '1' for 12.5 us  & '0' for 12.5 us error 3.25us ;
'0': T = 25 us, A = 1%;
```

[Tari = 6.25μs]

```
'1': Sig = '1' for 9.375 us  & '0' for 3.125 us error 1.56us;
'1': T = 12.5 us; A = 1%;
```

[Tari = 12.5μs]

```
'1': Sig = '1' for 18.75 us & '0' for 6.25 us error 3.125us;
'1': T =  25 us; A = 1%;
```

[Tari = 25μs]

```
'1': Sig = '1' for 37.5 us & '0' for 12.5 us error 6.25us;
'1': T = 50 us; A = 1%;
```

Figure 58: Textual description of Pulse-Interval Encoding (PIE).

```
'0': if Lprev = '0' then Sig ='1' for 3.1 us & '0' for 3.1 us;
     if Lprev = '1' then Sig ='0' for 3.1 us & '1' for 3.1 us;
'0': T = 6.2 us;  A = 7%;

'0': if Lprev = '0' then Sig ='1' for 2 us & '0' for 2 us;
     if Lprev = '1' then Sig ='0' for 2 us & '1' for 2 us;
'0': T = 3.9 us;  A = 10%;

'0': if Lprev = '0' then Sig ='1' for 1.5 ns & '0' for 1.5 us;
     if Lprev = '1' then Sig ='0' for 1.5 ns & '1' for 1.5 us;
'0': T = 3.1 us;  A = 10%;

'0': if Lprev = '0' then Sig ='1' for 0.7 ns & '0' for 0.7 us;
     if Lprev = '1' then Sig ='0' for 0.7 ns & '1' for 0.7 us;
'0': T = 1.5 us;  A = 15%;

'1': if Lprev = '0' then  Sig ='1' for 6.2 us;
     if Lprev = '1' then  Sig ='0' for 6.2 us;
'1': T = 6.2 us;  A = 7%;

'1': if Lprev = '0' then Sig ='1' for 3.9 us;
     if Lprev = '1' then Sig ='0' for 3.9 us;
'1': T = 3.9 us;  A = 10%;

'1': if Lprev = '0' then Sig ='1' for 3.1 us;
     if Lprev = '1' then Sig ='0' for 3.1 us;
'1': T = 3.1 us;  A = 10%

'1': if Lprev = '0' then Sig ='1' for 1.5 us;
     if Lprev = '1' then Sig ='0' for 1.5 us;
'1': T = 1.5 us;  A = 15%;
```

Figure 59: Textual description of FM0 Encoding.

6.3.1.4 FM0 The FM0 encoding scheme is one of the encodings used in the ISO 18000 Part 6C. It is used for backscattering the passive tag response, through absorption or reflection of the transmitted RF energy. The waveform of FM0 is shown in Figure 57(b). FM0 encoding determines its value by examining the level in its bit window. As shown in Figure 57(b), unlike Differential Manchester Encoding if an transition occurs in the middle of the bit window, this encodes a bit-'0'. If there is no transition in the bit window, this encodes a bit-'1'. In addition, FM0 requires

inverting level at every symbol (e.g. bit window) boundary. Therefore, the starting level of each bit encoding is based on the ending level of the previous bit. For example, “01” encoding can be represented by either “high-low” followed by “high” or “low-high” followed by “low”.

The textual representation of FM0 is shown in Figure 59. The description of FM0 is similar to that of Differential Manchester except that each bit value has four different representations corresponding to each of the four data rates. Also the encoding for ’1’ is simpler, as it does not have a transition within the bit window and as such has no & in the waveform description.

The physical layer encodings for FM0 are shown in Figure 57(b). Unlike PIE, FM0 is a fixed period encoding. The data rate of FM0 can be one of several discrete values as specified by the ISO 18000 Part 6C standard. These values include 160, 256, 320, and 640 kbps. The corresponding bit window periods are 6.2, 3.9, 3.1, and 1.5 μs with an error tolerance of 7%, 10%, 10%, and 15%, respectively. To encode a ’0’, there must be a transition at the middle of a bit window. To encode a ’1’ there is no transition within a bit window. However, between two adjacent bits there must be a transition at the edge of the bit window.

6.3.1.5 Modified Miller Encoding

Modified Miller encoding [107] is an encoding scheme that is often employed in near field communication, or communication of 10 cm or less [108]. Modified Miller encoding has a low pulse either at the beginning of the bit window to encode a ’0’, or delayed by half a period to encode a ’1’. However, if a ’0’ is proceeded by a ’1’ the ’0’ is encoded with no low pulse at all. This is shown in Figure 60.

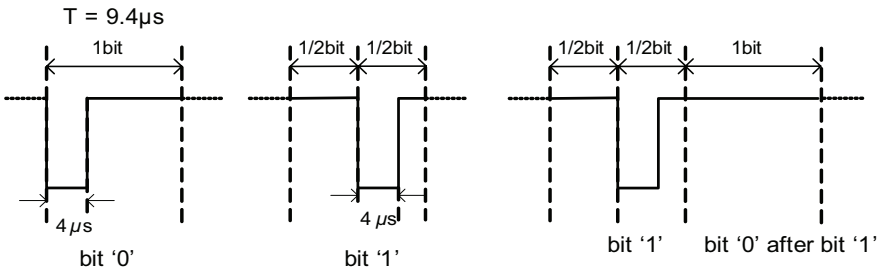


Figure 60: Continuous waveform for bits ’0’ and ’1’ of Modified Miller encoding.

To represent this in the waveform description text file, we introduce a new field, `Vprev`, which corresponds to the previously encoded value. In the example from Figure 61, the modified Miller encoding is shown for a period of 9.4 μs. To encode a '0' we either see a pulse at the beginning of the window if the current bit was proceeded by a '0', or no pulse if it was proceeded by a '1'. The encoding for '1' does not specify a `Vprev` value.

```
'0': if Vprev = '0' then Sig = '0' for 4 us & '1' for5. 4 us;
     if Vprev = '1' then Sig = '1' for 9.4 us;
T = 9.4 us;  A = 5%;

'1': Sig = '1' for 4.7 us & '0' for 4 us & '1' for 0.7 us;
T = 9.4 us; A = 5%;
```

Figure 61: Textual description of Modified Miller Encoding.

The preamble in an RFID packet alerts the system that a transmission packet is beginning. It typically includes a sequence of several pulses that are different from the encoded values in the encoding. The preamble must be matched exactly before any data transmission can commence. Figure 62 shows an example preamble using a textual *preamble description* starting with a 15 μs pulse followed by 5 μs pulses separated by 10 μs. A typical preamble could continue for several more pulses.

```
preamble: pre = '1' for 15 us & '0' for 15 us & '1' for 15 us
          & '0' for 15 us & '1' for 15 us & '0' for 15 us....;
```

Figure 62: Preamble textual representation example.

RFID standards may also specify transmission protocols between readers and tags. Thus, corresponding *transmission characteristics* must be declared. For example, the transmission protocol defined in ISO 18000 Part 7 specifies that an RFID reader transmits the least significant bit (LSb) first within a byte, and sends the most significant byte (MSB) first within a packet. Each byte is followed by a stop bit within a packet. The transmission order, MSB or LSB, determines the order of receiving data, and sending a response.

The textual description of each protocol ends with transmission characteristics. For example, the characteristics for ISO 18000 Part 7 are shown in Figure 63. This declares that in a byte the LSb is first, in a packet the MSB is first, each byte contains nine bits and one of these bits is a stop bit. Thus, a complete physical layer description file contains a waveform description, preamble description, and a transmission characteristics description.

```
transmission: bitOrder = least; byteOrder = most;
              byteSize = 9; stopBits = 1;
```

Figure 63: Serial transmission characteristics

6.3.2 Parse Procedure of Waveform Features

Each encoding has its own waveform properties. Manchester encoding requires a transition in the middle of each fixed-size bit window. Differential Manchester encoding determines the bit value by examining the transistion at the boundary of each bit window. PIE encoding uses different sized bit windows to determine the bit value. FM0 encoding requires a transition at the boundary of each bit window. Modified Miller encoding contains multiple transitions within one bit window. Table 13 lists a summary of waveform features for five encodings.

Table 13: The summary of waveform features for five encodings.

Encodings	Number of Transitions	Bit Window		Dependency		Bit Period (μs)	Accuracy (%)	Error (%)
		Fixed	Floating	Level	Value			
M[2]	1	X				U/D[5]	U/D	U/D
DM[3]	1	X		X		U/D	U/D	U/D
PIE	1		X			U/D	U/D	U/D
FM0	0/1	X		X		U/D	U/D	U/D
MM[4]	0/1/2	X			X	U/D	U/D	U/D

[2]M is the abbreviation of Manchester encoding.
[3]DM is the abbreviation of Differential Manchester encoding.
[4]MM is the abbreviation of Modified Miller encoding.
[5]U/D is the abbreviation of User Defined.

The *RFID System Class*, declared as the top-level class, contains two private classes, *Signal Class* and *Preamble Class*, as shown in Figure 64. The *RFID System Class* stores waveform features captured by the parse function call, such as the "Number of Transitions", "Bit Window", "Dependency", "Bit Period", "Accuracy", and "Error" . The *Signal Class* records the signal encoding information and the *Preamble Class* records the preamle encoding information.

The *Signal Class* uses a **List** data structure to contain encoding information for each bit value. As an example in Figure 64, the *Signal Class* contains `Bit-'0'` and `Bit-'1'` signal classes. Each *Bit-value Signal Class* is composed of a **List** of type *Signal Feature Class*. The *Signal Feature Class* is a parent class of *Level-feature* and *Edge-feature* classes.

Figure 65 indicates how to develop a **List** of type *Signal Feature Class* from a textual waveform description. For instance, `'0': Sig = '1' for 18 us & '0' for 18 us` describes the encoding scheme for bit-'0' of Manchester encoding, as an example in Figure 65(a). This waveform description is parsed into three signal features: **level-high**, **edge-falling**, and **level-low** features. This feature list is developed for the *Bit-'0' Signal Class*. The *Bit-'1' Signal Class* is constructed by using the same approach as shown in Figure 65(b). It is composed of three signal features: **level-low**, **edge-rising**, and **level-high** features. Finally, the *RFID System Class* contains detailed information on the encoding schemes which are utilized to generate encoder/decoder VHDL code.

Each *Signal Feature Class* includes details on encoding information such as level value and timing requirements. For example in Figure 65(a), Feature 1 contains the encoding information such that **Level(High)** for **18**μs with an accuracy of **12.5**%. Feature 2 contains the encoding information such as: the **Edge(Falling)** transition occurring at **18**μs with an accuracy of **12.5**%.

6.3.3 Waveform Feature Library and VHDL Generation

The physical layer waveform feature library in Figure 51 is a collection of basic hardware-based components corresponding to various waveform features. For example, this set contains a hardware based edge detector, a sampling counter, sampling registers, serial-to-parallel hardware blocks, first-in-first-out (FIFO) blocks, and other basic blocks as predefined components in the library. These components are programmable and designed to fit different user defined parameters specified

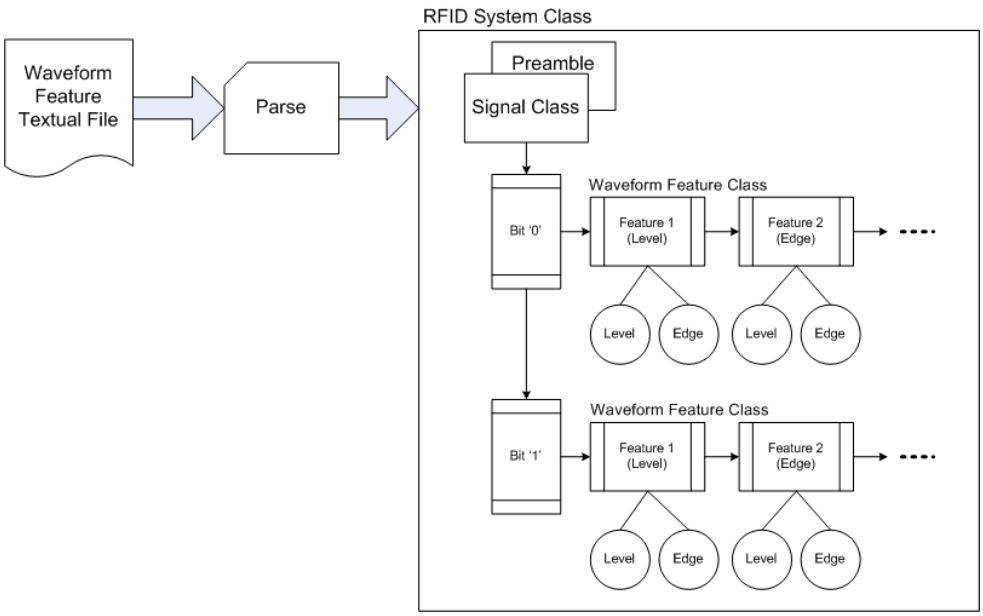
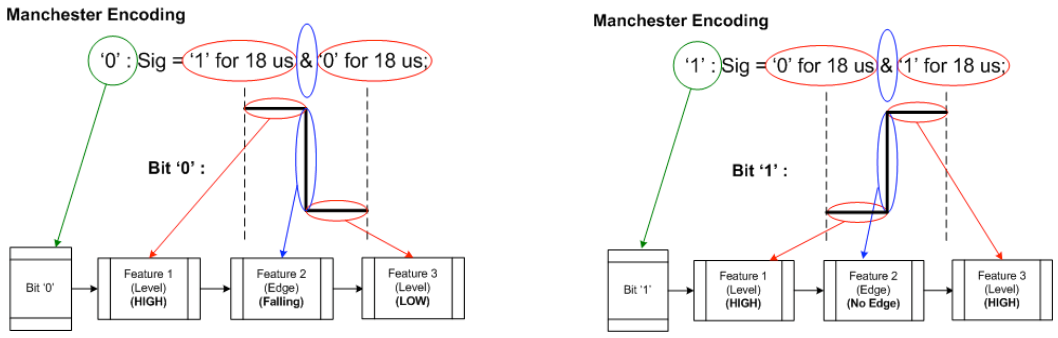


Figure 64: Conceptual diagram of the *PARSE* procedure.

in the textual description from Section 6.3.1. A hardware-based edge detector for instance, can be used for detecting only a raising edge, only a falling edge, or either edge. Similarly, a sampling counter can vary based on sampling rates corresponding to the different data toggling rates and/or different duty cycles of bit data in the description.

Waveform variables are parameters that are captured from the waveform description file. These parameters are used to describe the physical layer characteristics of an encoding mechanism in a way that the the synthesis process can understand. Thus, the synthesis process translates the textual description into the parameters for the feature library and maps the parameters to the correct hardware component.

A bit window period is a simple example of a waveform characteristic. A bit window can contain a transition at a particular time during the period. Another characteristic is the direction of this transition. From Figure 54(a), the Manchester encoding requires a 50% duty cycle waveform with a period of 36 μs where the direction determines the value. A '1' has identical features to a '0' except that it is composed of a rising edge at the middle of a bit window as opposed to a falling



(a) Bit '0' of Manchester encoding.

(b) Bit '1' of Manchester encoding.

Figure 65: Developing a feature list of *bit-'0' signal class* and *bit-'1' signal class*.

edge. Because the waveform is a continuous wave, a transition may also occur at the edge of two adjacent bits depending on the values of these windows.

The bit rate for a waveform with a period of 36 μs is 27.7 kbps. Because of the 50% duty cycle, a transition occurs at the middle of a bit window and logic levels may change between bit windows. Therefore, the data signal toggling rate is 55.4kbps. The sampling rate (f_s) must be at least two times faster than the toggling frequency (f_t) of the target signal: $f_s = 2 \times f_t$. Since the data signal toggling rate is 55.4kbps (f_t) the minimum sampling rate is 110.8kbps, or four times oversampling. However, the `A` parameter of the description tells us the typical fluctuation that might occur in the signal, which was specified as 12.5%. Thus, a minimum of eight times oversampling should be used for edge detection and synchronization.

An overview of the general waveform detection circuit is shown in Figure 66. This circuit contains a preamble detection circuit, an edge detection circuit used for synchronization with the incoming waveform, a timer circuit for signaling the controlling state machine to change states, a sampling register file for converting levels and edges into decoded bit values, and a serial to parallel converter for building bytes from the incoming bits. The system clock speed is also variable based on the required sampling by the circuit.

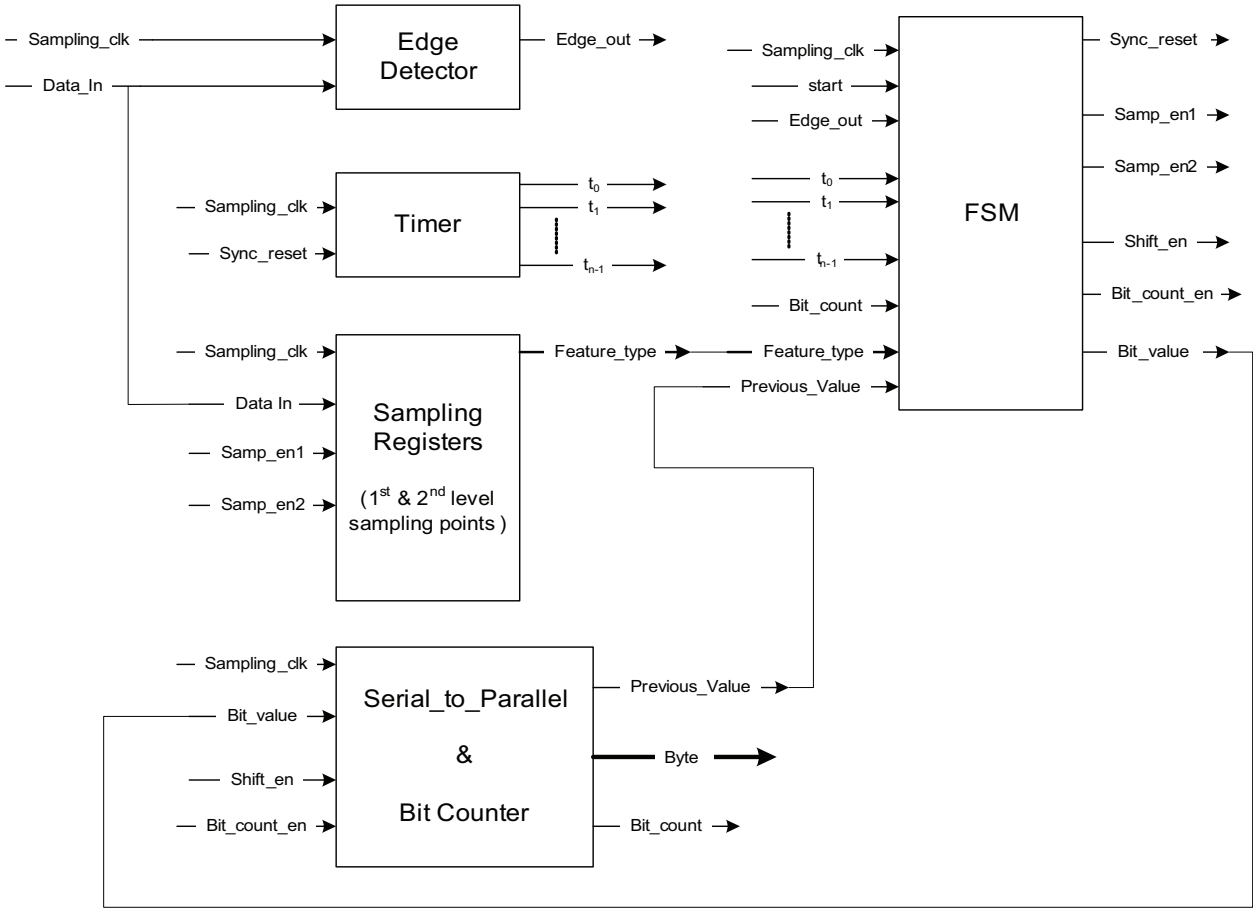


Figure 66: General waveform detection circuit.

The data transmission begins after a valid preamble is detected by the `start` signal. The *edge detection circuit* shown in Figure 67 is used to keep in synchronization with the incoming waveform. For every system cycle in which the value changes, it signals the timer that an edge has occurred.

The *timer circuit* shown in Figure 68 controls the sampling times used to check for level or value changes in the incoming waveform. The timer circuit contains counters that count system cycles until the next sampling window. Whenever the timer matches the time point, the system samples the value of the decoded data for the further comparison. Thus, the hardware library

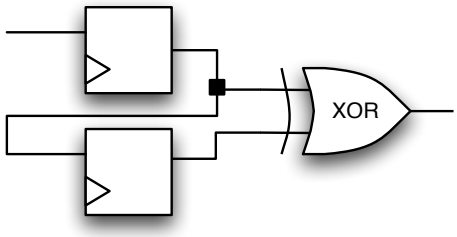


Figure 67: Edge detection circuit.

contains parameters for the number of time points, n, to sample, the number of cycles to count between each sample, N_0, N_1, ..., N_{n-1},and the bit width of the counter which is $log_2 \max(N)$.

Recall the example in Figure 65(a). Feature 1 contains the encoding information such as: **Level(High)** for **18**μs with an accuracy of **12.5**%. In this example, four time points are needed: (t_1, t_2) represents the period of level-high, and (t_3, t_4) represents the range of the accuracy. For the edge transition example, Feature 2 in Figure 65(a) contains the encoding information such as: **Edge(Falling)** transition occurring at **18**μs with an accuracy of **12.5**%. In this edge-feature example, Feature 2 needs two time points: (t_5, t_6) represents the range of the falling transition.

The timer signals become inputs into an automatically generated finite state machine (FSM) *controller circuit*. The FSM uses the timer signals and the sampling registers block to determine the actual encoded values. The *sampling registers circuit*, shown in Figure 69, samples two data values when signaled by the controller and reports back whether there has been a rising edge ('0' followed by a '1'), a falling edge ('1' followed by a '0'), or a constant level ('1' or '0'). Based on these values, the controller traverses states to match one of the bit conditions specified in the textual description. It can also base this on one or more previously seen values seen in the serial stream with the `previous_value` signal.

The controller uses the synchronization signal `Edge_out` to tell the timer when to reset with the `Sync_reset` signal. For cases where the bit window is not fixed, a bit may be determined before all timers have expired as with PIE encoded values of '0.' The controller can reset the timer early in these cases to begin looking for the next bit.

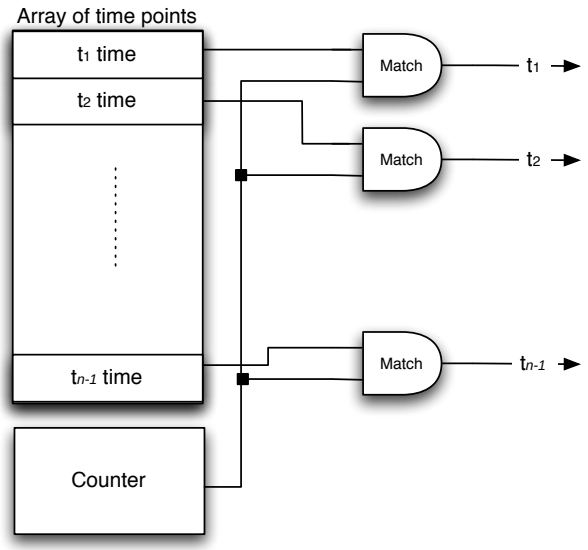


Figure 68: Timer circuit schematic.

Finally, once a bit is determined it is fed into the *serial to parallel* circuit. This circuit, shown in Figure 70, has parameters such as the number of bits per byte and the direction of shifting. For MSb first the circuit shifts bits into the register to the left and for LSb first the circuit shifts bits to the right. When buffering the whole packet a similar shifting technique is used for most and least significant bytes (MSB and LSB).

6.3.4 Synthesis Process and VHDL Generation

The synthesis process from the textual description is primarily based on discovering sampling points based on the waveform properties. For example consider the modified Miller encoding from Figure 60. The basic waveforms for '0' and '1' with an inverse pulse indicate sampling both during the pulse and outside the pulse. The synthesis process first selects sampling points in the center of a level region, thus at 2 μs and 6.7 μs for a '0' and at 2.7 μs, 6.7 μs, and 9.05 μs for a '1.' First, the 9.05 μs is discarded, because the signal is high for all three descriptions. 6.7 μs is retained directly, because it matches both the '1' and '0' directly. The 2 μs and 2.7 μs values are determined to represent the same sampling window as they are within stable regions for both

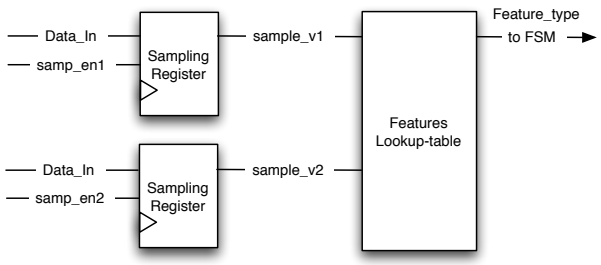


Figure 69: Sampling registers circuit schematic.

the '1' and '0' waveforms. As a result, a value within the range of 2 to 2.7 may be selected for a sampling point.

The controller FSM is generated to detect different sequences of features for each encoded bit in a manner similar as a numeric sequence detector. For example, if there is a rising edge between the samples, it is a '0,' a falling edge indicates a '1,' and a constant level high is a '0.' Both versions of '0' can be checked against the previous value.

For a PIE encoding (Figure 57(a)) there are three sampling points to consider. This is demonstrated for the first description of each '0' and '1' from Figure 58. According to the description, a falling edge occurs between 2.9 and 4.2 μs making this an "invalid detection region." Thus for a '0' the sampling points are 1.45 μs, and 5.225 μs. Because the period is not fixed for a '1' and the "invalid detection region" and period completion time overlap, the system can move into active sampling mode. Because a falling edge has not occurred by 4.2 μs, one must occur between 6.1 and 10.5 μs. A timer indicates when 6.1 μs have elapsed, then the FSM looks for an edge before the timer indicates 10.5 μs. Finally, upon seeing a second edge, the FSM begins a new bit window.

The process for encoding these values appropriately is much simpler than the detection process. The process requires a *parallel to serial* block complementary to the *serial to parallel* block. Each bit waveform is generated with a very simple controller FSM that uses timers to traverse states, with each state outputting one particular level. This is a fairly straightforward conversion process from the textual representation.

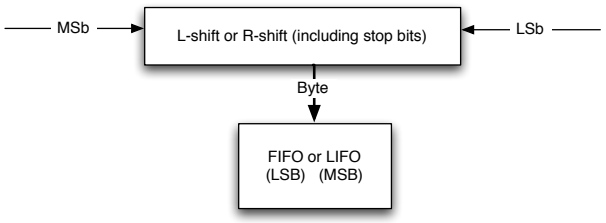


Figure 70: Serial-to-parallel circuit schematic.

VHDL code generation is the final phase of the synthesis flow. The structure of the block is shown in Figure 66. Several of the included libraries are parameterized with VHDL `generic` constructs for specified parameters including the timer and serial to parallel blocks. The FSM controller is entirely generated by the synthesis engine using a `generic` for the number of timer signals to include.

6.4 RESULTS

Hardware was generated for the five encodings; Manchester, differential Manchester, PIE, FM0, and modified Miller. We implemented these hardware blocks on several different reprogrammable targets we use for RFID prototyping including a Xilinx Spartan III 400 FPGA, an Actel Fusion AFS90 flash-based FPGA, and a Xilinx Coolrunner II CPLD. The Spartan III was selected because we have a credit card sized prototyping board with this device for demonstration. The Fusion was selected because it is relatively low power and contains non-volatile programming bits and memory for passive prototyping. The CPLD was selected because of its favorable power properties. We also synthesized these designs into standard cell ASIC netlists for comparison.

Performance and area estimates were made using post synthesis, placed and routed netlist simulations of the designs. Power estimation was completed using stimuli from the simulation with vendor supplied tools. For the ASIC implementation area and performance estimates are from Design Compiler. The cell library used for synthesizing the design is 0.16 μm technology. Power estimates are from post synthesis simulation using Mentor Graphics Modelsim and analysis

using Synopsys Primepower. Results for the physical layer synthesis flow are shown Table 14 with manual results shown in parenthesis for comparison. Details on the encoders have been omitted because their hardware is trivial compared to that of the decoders in each case. Based on our studies, in terms of area and power consumption, the automatically generated designs are very close to manually generated designs and in most cases are more efficient.

Table 14: Statistics of decoder blocks with the automatically generated value followed by manual design value.

	Man	**Diff Man**	**PIE**	**FM0**	**MM**
Freq	1 MHz	1 MHz	10 MHz	4 MHz	10 MHz
Xilinx Spartan 3 XC3S400: area in cells, power in mW					
Area	108 (108)	103 (98)	192 (216)	109 (113)	119 (105)
Power	0.02 (0.02)	0.02 (0.01)	0.31 (0.39)	0.10 (0.12)	0.25 (0.31)
Altera Fusion AFS90: area in cells, power in mW					
Area	140 (153)	146 (206)	276 (306)	151 (159)	159 (142)
Power	0.10 (0.18)	0.11 (0.20)	1.25 (1.02)	0.40 (0.51)	1.06 (1.10)
Xilinx Coolrunner II 256: area in gates , power in mW					
Area	314 (310)	329 (293)	625 (586)	335 (315)	372 (356)
Power	0.18 (0.18)	0.19 (0.18)	1.98 (2.13)	0.65 (0.62)	1.62 (1.53)
Oki 0.16μm cell-based ASIC: area in μm^2, power in μW					
Area	3585 (4128)	3998 (4323)	10713 (13414)	5134 (6058)	7299 (7218)
Power	0.61 (0.47)	0.63 (0.49)	4.31 (3.98)	1.74 (1.94)	3.71 (4.51)

The Manchester and Differential Manchester decoders have similar architectures. The size and power consumption are very close. These decoders also require relatively low sampling rates of 1, 4, and 10 MHz. The modified Miller encoding requires a similar area to Manchester encoding but requires considerably more power. This is due to the ten time higher sampling frequency. FM0 and PIE encoding requires significantly higher area (2-3 times) more than the other encodings. Due to the increased sampling rate and complexity, these encodings require significantly more power consumption.

Based on our studies, in general, the automatically generated decoding designs require 20 %less in gate/cell than hand generated designs. However, simplified state machines (decoder control units) of the automatically generated decoding designs cause more energy consumption. According to our experimental results, the automatically generated decoding designs consume 7%

more than normal hand generated designs. As a result, the FPGAs provide a 0.1 to 2mW level solution, while the ASIC provides a 0.6 to 5μW level solution for decoders generated by our Physical Layer Synthesis tool.

6.5 CONCLUSION

This chapter presents a physical layer design automation flow intended for RFID tags. The RFID Physical Layer Generator automatically generates physical layer encoding and decoding blocks in hardware through the description of *waveform features* of the encoding. We have implemented and compared five different physical layer encodings, Manchester, differential Manchester, PIE, FM0, and modified Miller encodings.

7.0 CONCLUSION AND FUTURE DIRECTIONS

7.1 PROBLEM STATEMENT

Active RFID technology has several challenges related to energy dissipation, inadequate security, system verification capabilities, and design cost. They are enumerated as follows:

1. **Unnecessary Power Dissipation:** Power is dissipated in a traditional active RFID tag to many factors such as an always-on active transmitter and tag controller. If the power dissipation can be reduced, battery life of the tag can be extended or contain additional capabilities, such as data encryption. It is important to develop a new technique to assist active RFID tags to decrease power consumption.
2. **The Need of Low-power Security Features:** Traditional active RFID tags provide security features by through their processor-based controller which increases the power consumption overhead. This situation creates a security strength versus energy trade-off. However, RFID systems require security features to be implemented using techniques that provide a high degree of protection while not significantly increasing the complexity of the system as complexity can increase power consumption and implementation cost. Thus, it is important to develop new security techniques that take advantage of the fundamental properties of RFID communication such as physical layer protocols, low-power communication extensions, sleep modes, physical implementation variations, etc.
3. **A Verification Platform for RFID Systems:** RFID technology needs a verification technique for the network traffic of RFID systems. A computer network debugging tool, *tcpdump* as an example, allows the user to intercept TCP/IP packets and display them. Upon the collected packet information, the user is capable of verifying and analyzing the network, monitoring the

traffic, and even providing a protection to the network. However, there is no tool like *tcpdump* in RFID systems, which provides detailed network traffic of RFID communications from the system level perspective. It is essential to develop a new verification platform which provides a system-level verification for RFID systems.

4. **Increasing Design Cost and Time:** RFID technology confronts the challenges of the increasing cost for designing and implementing a variety of RFID systems. In order to meet the requirements for different RFID circumstances and support extensive RFID applications, multiple standards exist. In most applications, RFID tag and reader hardware/software must be specifically designed for each application. This keeps overall design time long and the system cost high. It is essential to develop a design automation tool for reducing the design cost and design time.

7.2 CONCLUSION

This dissertation provides a multi-domain solution to improve the power consumption and security, while also reducing design time and verification time of RFID systems. In particular, I describe (1) a smart buffering technique to allow a tag to remain in a standby mode until addressed, (2) a multi-layer, low-power technique that transcends the passive-transaction, physical, and data layers to provide secure transactions, (3) an FPGA-based traffic profiler system to generate traces of RFID communications for both tag verification and power analysis without the need for actual hardware, and (4) a design automation technique to create layer encoding and decoding blocks in hardware suitable for RFID tags.

This dissertation presents four contributions for these four problems: (1) The *Smart Buffer* technique is designed and developed to avoid the waste of the power dissipation. As a result, based on a Markov Process energy model, the smart buffering technique is shown to reduce power consumption by 82%, 88%, and 90% over a traditional active tag; (2) The multi-layer, low-power security technique provides protection against malicious reader attacks to disable the tag, to steal the information stored in or communicated to the device. It provides security features, such as authentication and encryption mechanisms, without increasing the complexity of the tag system

too much. The power consumption overhead for implementing these layers of security is increased approximately 13% over the basic tag controller; (3) In addition, the FPGA-based traffic profiler system has been developed and implemented as the new verification platform for RFID systems. It is capable of generating traces for ISO 18000 part 6C (EPC Gen2) protocol. The trace files provide insight information of RFID communication transactions; and (4) The Physical Layer Design Automation tool is developed and implemented to generate encoding/decoding hardware blocks automatically. It is capable of reducing the design cost and time. This tool has been verified with five data-encoding protocols used in or related to RFID systems. Consequently, any power consumption of five designs is less than 5 μW. Furthermore, compared with five designs implemented by hand, the difference of the power consumption between two of them is less than 7% at most.

7.2.1 Primary Contributions

Detailed conclusions of four contributions are enumerated as follows:

1. **Smart Buffer Technique:** This dissertation proposed the *Smart Buffer* technique, and its architecture. The *Smart Buffer* allows a tag to remain in a sleep/standby mode until addressed. This chapter also illustrates a RFID testing circumstance in a testing scenario and an associated Markov Process model to analyze the efficiency of power saving.

   Two observations are made that energy is wasted mainly when (1) irrelevant point-to-point commands are sent to a group of tags, and when (2) the receiver of each tag always listens to the incoming signals. Based on these observations, the *Smart Buffer* technique is proposed to assist power saving for a passive active RFID tags (PARTs). The *Smart Buffer* is capable of filtering irrelevant commands and remaining the microcontroller in sleep mode until it is addressed. Three testing scenarios have been built for testing the power efficiency for a tag with or without the *Smart Buffer* technique. In the scenario one, an interrogator only send one point-to-point command to every tag. In the scenario two and three, an interrogator only send 50 and 100 point-to-point commands to every tag. Our experiments show that the average energy saved by adding the *Smart Buffer* is 82%, 88%, and 90% per tag where $N = 100$ for scenarios one, two, and three, respectively.

2. **Multi-layer Low-power Security:** The multi-layer, low-power security technique provides advanced secure transactions on the passive-transaction, physical, and data layers. The experimental results demonstrate the implementation of our encryption algorithm has resistance to the side-channel attacks.

   A secure transmission methodology for PART is presented where our primary concern is functionality and power (battery lifetime). Our technique provides layers of security as multiple levels of defense against attack. The passive activation layer is encoded to prevent a malicious reader from continuously activating the tag to drain its battery or to attempt to issue malicious commands. This layer of security provides an authentication mechanism for tag systems. The physical layer is encrypted using a mixture of Manchester and Differential Manchester encoding. Finally, the data itself is encrypted using the AES implemented to avoid the key from being broken using DPA. These two layers of security provide the encryption mechanisms for tag systems.

   These security modifications are implemented with a minimal power overhead from the actual RFID transaction power required. The burst switch detection circuit requires $< 1\mu$W when operating at 100 kHz even for 32-bit values ($.58\mu$W). The modified Manchester, Differential Manchester decoder requires approximately 1μW of additional power (1.11μW). Finally, the AES operation requires a maximum of approximately 7μW at 10 kHz (7.031μW), which is more than sufficient for the data transmission speed. Based on the active tag primitive logic controller implemented in the same hardware, depending on the number of primitives included (up to 40) the power consumed is 63-65 μW [86]. Thus, the overhead for the security is just under 9μW (8.721 μW) which is 13.4% increase over the active tag logic alone.

3. **FPGA-based Traffic Profiler:** The FPGA-based traffic profiler system generates traces of RFID communications for both tag verification and power analysis without the need of actual hardware. It demonstrates how to use the generated traffic trace file to profile the power consumption of the communication.

   Two case studies illustrate the usage of the trace file generated by the RFID traffic profile system. Based on the trace files, a user is capable of understanding the details on the communication transaction and verifying the RFID network traffic. The first case study illustrates the method of analyzing the RFID network. For example, in order to study the collisions be-

tween tags, it is important to understand the information of the slot counter from tags. Every tag has its own methodology to pick the slot counter or to generate the slot counter based on the Q value. However, there is no way to obtain the slot counter value by using a traditional network analyzer or spectrum analyzer. The RFID traffic profile system provides an alternative approach to obtain this information for the users.

The second case illustrates the power estimation for the RFID communication by utilizing the RFID traffic profile system. By sending "auto read" command from the SAMSys reader for two seconds, a tag needs to consume 2.765 μJ for processing `Query`, 0.471 μJ for processing `QueryAdj`, 0.796 μJ for processing `QueryRep`, 0.047 μJ for processing `Nak`, and 0.421 μJ for processing `Ack`.

4. **Physical Layer Design Automation Tool:** The physical layer design automation tool is the design automation technique to generate encoding and decoding hardware blocks automatically for the physical layer data encodings. The main benefit of creating this tool is to shorten the design time for a variety of RFID standards and applications.

   This tool has been verified with five data-encoding protocols used in or related to RFID systems. They are the Manchester, the Differential Manchester, FM0, Pulse-Interval Encoding (PIE), and Modified Miller encodings.

   Based on our studies, in general, the automatically generated decoding designs require 20 %less in gate/cell than hand generated designs. However, simplified state machines (decoder control units) of the automatically generated decoding designs cause more energy consumption. According to our experimental results, the automatically generated decoding designs consume 7% more than normal hand generated designs. As a result, the FPGAs provide a 0.1 to 2mW level solution, while the ASIC provides a 0.6 to 5μW level solution for decoders generated by our Physical Layer Synthesis tool. Due to the increased sampling rate and complexity, these encodings require significantly more power consumption.

7.3 FUTURE DIRECTIONS

Future directions of the dissertation include *security architecture for RFID systems*, *verification platform of RFID systems*, and *integration with RFID compiler*.

7.3.1 Security Architecture for RFID Systems

The multi-layer low-power security provides an advanced low-power security mechanism for RFID systems. This technique is designed and developed from a hardware perspective. A complete security architecture is needed for RFID systems to cover not only the hardware portion, but also the software implementation. For example, a key distribution service is a good direction to be concerned. Furthermore, adding security primitives to RFID basic primitives for RFID compiler might be another research challenge. A future direction of extending the research of the multi-layer low-power security will be a demonstratable prototype of security architecture for RFID systems.

7.3.2 Verification Platform of RFID Systems

The FPGA-based traffic profile system proposed in this dissertation is capable of verifying RFID communication transactions. Since it is designed on an FPGA device, the hardware-portion design can be reloaded based on the "**System Under Verification**", which might be different RFID standard for different testing case. A verification platform for either a single RFID system or multiple RFID systems is an interesting direction.

7.3.3 Integration with RFID Compiler

The physical layer design automation tool is capable of automatically generating physical encoding/decoding block in hardware. The RFID compiler proposed in Dontharaju's dissertation [102] is capable of automatically generating a hardware-based RFID controller based on the user's script of RFID specification. It is an interesting research topic to develop a new RFID compiler integrating with the physical layer design automation tool. Such that the new RFID compiler is capable of automatically generating the whole chip for an RFID tag when the users only provide the script of RFID specifications and the description of waveform features.

BIBLIOGRAPHY

[1] klaus Finkenzeller, *RFID Handbook - Fundamentals and Applications in Contactless Smart Cards and Identification*. Kluwer Academic Publishers, 2003.

[2] D. Paret, *RFID and Contactless Smart Card Applications*. John Wiles & Sons, 2005.

[3] International Standards Organization, "ISO/IEC FDIS 18000-7:2004(E)." Standard Specification, 2004.

[4] American National Standards Institute, "ANSI NCITS 236:2001." Standard Specification, 2002.

[5] S. C. A. I. Council, "RF-Enabled Applications and Technology: Comparing and Contrasting RFID and RF-Enabled Smart Cards," Tech. Rep., Smart Card Alliance, Jan. 2007.

[6] M. C. O'Connor, "Homeland Security to Test RFID." RFID Journal, 2005. http://www.rfidjournal.com/article/articleview/1360/.

[7] E. Chabrow, "Homeland Security To Test RFID Tags At U.S. Borders." InformationWeek.com, 2005. http://www.informationweek.com/story/showArticle.jhtml? articleID=57703738.

[8] S. E. Sarma, "Towards the Five-Cent Tag," Tech. Rep. MIT-AUTOID-WH-006, Massachusetts Intitute of Technology, 2001.

[9] R. Roman, C. Alcaraz, and J. Lopez, "A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes," *Mob. Netw. Appl.*, Vol. 12, No. 4, pp. 231–244, 2007.

[10] H. Cho and Y. Baek, "Design and Implementation of an Active RFID System Platform," *SAINT-W '06: Proceedings of the International Symposium on Applications on Internet Workshops*, (Washington, DC, USA), pp. 80–83, IEEE Computer Society, 2006.

[11] H.-J. Chae, D. J. Yeager, J. R. Smith, and K. Fu, "Maximalist cryptography and computation on the WISP UHF RFID tag," *In Proceedings of the Conference on RFID Security*, July 2007.

[12] M. Feldhofer, S. Mominikus, and J. Wolkerstorfer, "Strong Authentication for RFID Systems using the AES algorithm," *Proc. Of CHES 2004*, Vol. 3156 of *Lecture Notes on Computer Science*, pp. 357–370, 2004.

[13] G. D.Bhanage, Y. Zhang, Y. Zhang, W. Trappe, and R. E. Howard, "RollCall : The Design For A Low-Cost And Power Efficient Active RFID Asset Tracking System," *EUROCON, 2007. The International Conference on "Computer as a Tool"*, pp. 2521–2528, Sept. 2007.

[14] G. Mazurek, "Collision-Resistant Transmission Scheme for Active RFID Systems," *EUROCON, 2007. The International Conference on "Computer as a Tool"*, pp. 2517–2520, Sept 2007.

[15] B. Zhen, M. Kobayashi, and M. Shimizu, "To read transmitter-only RFID tags with confidence," *Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium on*, pp. 396–400, Sept 2004.

[16] R. Das and P. Harrop, "RFID Forecasts, Players & Opportunities 2008-2018," Tech. Rep., IDTechEx.com, 2008.

[17] M. R. Rieback, B. Crispo, and A. S. Tanenbaum, "The evolution of RFID security," *Pervasive Computing, IEEE*, Vol. 5, No. 1, pp. 62–69, 2006.

[18] D. Boys, "Identification Friend or Foe IFF Systems: IFF Questions & Answers," 2005. www.dean-boys.com/extras/iff/iffqa.html.

[19] B. Manish and M. Shahram, *RFID Field Guide: Deploying Radio Frequency Identification Systems*. Prentice Hall, 2005.

[20] G. Veeder-Root, "RFID Cashless/Wireless Payment." http://www.gilbarco.com/pdfs/P2319.pdf.

[21] T. A. F., "Machine Readable Travel Documents," Tech. Rep. 1.1, International Civil Aviation Organization, 2004. PKI for Machine Readable Travel Documents offering ICC Read-Only Access.

[22] International Standards Organization, "ISO/IEC FDIS 18000-6:2004/Amd 1:2006(E)." Standard Specification, 2006.

[23] EPCglobal, Inc., *EPC Radio-Frequency Identity Protocols: Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz*, version 1.0.9 ed., January 2005.

[24] International Standards Organization, "ISO/IEC FDIS 14443, Proximity cards (PICCs)." Standard Specification, 2000.

[25] L. Sullivan, "DOD Seeks New Active-Tag Suppliers." RFID Journal, 2006. http://www.rfidjournal.com/article/articleview/2856/1/1/.

[26] B. Nilsson, L. Bengtsson, U. Bilstrup, P.-A. Wiberg, and B. Svensson, "Towards an Energy Efficient Protocol for Active RFID," *Embedded Systems, 2006. IES '06. International Symposium on*, pp. 1–4, Oct. 2006.

[27] B. Nilsson, L. Bengtsson, P.-A. Wiberg, and B. Svensson, "Protocols for Active RFID - The Energy Consumption Aspect," *Industrial Embedded Systems, 2007. SIES '07. International Symposium on*, pp. 41–48, July 2007.

[28] G. Bhanage and Y. Zhang, "Relay MAC: A Collision Free And Power Efficient Reading Protocol For Active RFID Tags," *Computer Communications and Networks, 2006. ICCCN 2006. Proceedings.15th International Conference on*, pp. 97–102, Oct. 2006.

[29] T. Kaya and H. Koser, "A New Batteryless Active RFID System: Smart RFID," *RFID Eurasia, 2007 1st Annual*, pp. 1–4, Sept. 2007.

[30] J.-P. Chen, T.-H. Lin, and P. Huang, "On the potential of sensor-enhanced active RFIDs," *Emerging Information Technology Conference*, pp. 56–60, Aug. 2005.

[31] A. K. Jones, S. Dontharaju, S. Tung, P. J. Hawrylak, L. Mats, R. Hoare, J. T. Cain, and M. H. Mickle, "Passive active radio frequency identification tags (PART)," *International Journal of Radio Frequency Identification Technology and Application (IJRFITA)*, Vol. 1, No. 1, pp. 52 – 73, 2006.

[32] C. Grinstead and J. L. Snell, *Introduction to Probability*. American Mathematical Society, 1997.

[33] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. Richard D. Irwin, Inc., and Aksen Associates, Inc., 1993.

[34] R. O'Donnell, M. Harchol-Balter, and J. Lafferty, "Lecture 15: Discrete Time Markov Chains - Part A," 2006. 15-359 Probability and Computing Lecture.

[35] P. J. Hawrylak, *Analysis and Development Of A Mathematical Structure To Describe Energy Consumption Of Sensor Networks*. PhD thesis, University of Pittsburgh, 2006.

[36] A. K. Jones, R. R. Hoare, S. R. Dontharaju, S. Tung, R. Sprang, J. Fazekas, J. T. Cain, and M. H. Mickle, "A Field Programmable RFID Tag and Associated Design Flow," *Proc. of FCCM*, pp. 165–174, 2006.

[37] A. K. Jones, R. Hoare, S. Dontharaju, S. Tung, R. Sprang, J. Fazekas, J. T. Cain, and M. H. Mickle, "An Automated, FPGA-based Reconfigurable, Low-power RFID Tag," *Journal of Microprocessors and Microsystems*, Vol. 31, pp. 116–134, March 2007.

[38] S. E. Sarma, S. A. Weis, and D. Engels, "Radio-frequency-identification security risks and challenges," *CryptoBytes*, Vol. 6, No. 1, 2003.

[39] J. Mandel, A. Roach, and K. Winstein, "MIT Proximity Card Vulnerabilities," Tech. Rep., Massachusetts Institute of Technology, March 2004.

[40] S. Bono, M. Green, A. Stubblefield, A. Juels, A. Rubin, and M. Szydlo, "Security analysis of a cryptographically-enabled RFID device," *Proc. of the USENIX Security Symposium*, pp. 1–16, 2005.

[41] M. Burmester and B. de Medeiros, "RFID Security: Attacks, Countermeasures and Challenges," *Proc. of the RFID Academic Convocation, The RFID Journal Conference*, 2007.

[42] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler, "Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker.," *CHES*, pp. 101–118, 2006.

[43] K. Kfir and A. Wool, "Picking virtual pockets using relay attacks on contactless smartcard systems," *Proc. of SecureComm*, 2005.

[44] G. Hancke, "A Practical Relay Attack on ISO 14443 Proximity Cards," Tech. Rep., University of Cambridge, 2005.

[45] J. Coron, D. Naccache, and P. Kocher, "Statstics and Secret Leakage," *ACM Transactions on Embedded Computing Systems*, Vol. 3, pp. 492–508, August 2004.

[46] P. Kocher, "Timing Attacks on Implementations of Diffle-Hellman, RSA, DSS and Other Systems," *Proc. of the International Advances in Cryptology Conference (CRYPTO)*, pp. 104–113, 1996.

[47] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Proc. of the International Advances in Cryptology Conference (CRYPTO)*, Vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397, Springer-Verlag, 1999.

[48] E. D. M. P. B. S. O. P. D. B. P. G. V. I. Verbauwhede, "Electromagnetic Analysis Attack on an FPGA Implementation of an Elliptic Curve Cryptosystem," *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*, Vol. 2, pp. 1879–1882, 2005.

[49] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic Analysis: Concrete Results," *CHES '01: Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, (London, UK), pp. 251–261, Springer-Verlag, 2001.

[50] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully Attacking Masked AES Hardware Implementations," *Proc. of Cryptographic Hardware and Embedded Systems (CHES)* (J. R. Rao and B. Sunar, eds.), Vol. 3659 of *Lecture Notes in Computer Science*, pp. 157–171, Springer, 2005.

[51] D. D. Hwang, K. Tiri, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "AES-Based Security Coprocessor IC in 0.18-μm CMOS With Resistance to Differential Power Analysis Side-Channel Attacks," *IEEE Journal of Solid-State Circuits*, Vol. 41, pp. 781–792, April 2006.

[52] S. B. Örs, E. Oswald, and B. Preneel, "Power-Analysis Attacks on FPGAs – First Experimental Results," *Proc. of Cryptographic Hardware and Embedded Systems (CHES)* (C. D.

Walter, Çetin Kaya Koç, and C. Paar, eds.), Vol. 2779 of *Lecture Notes in Computer Science*, pp. 35–50, Springer, 2003.

[53] F.-X. Standaert, S. B. Örs, J.-J. Quisquater, and B. Preneel, "Power Analysis Attacks Against FPGA Implementations of the DES," *Proceedings of Field-Programmable Logic and its Applications*, Vol. 3203 of *Lecture Notes in Computer Science*, pp. 84–94, 2004.

[54] L. Benini, A. Macii, E. Macii, E. Omerbegovic, F. Pro, and M. Poncino, "Energy-aware design techniques for differential power analysis protection," *Proceedings of the Design Automation Conference (DAC)*, pp. 36–41, 2003.

[55] C. Gebotys, "A table masking countermeasure for low-energy secure embedded systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 14, pp. 740–753, July 2006.

[56] P. Rakers, L. Connell, T. Collins, and D. Russell, "Secure contactless smartcard ASIC with DPA protection," *IEEE Journal of Solid-State Circuits*, Vol. 36, pp. 59–565, March 2001.

[57] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining Smart-Card Security under the Threat of Power Analysis Attacks," *IEEE Trans. Comput.*, Vol. 51, No. 5, pp. 541–552, 2002.

[58] I. Vajda and L. Buttyan, "Lightweight Authentication Protocols for Low-Cost RFID Tags," *Proc. of Ubiquitious Computing (UBICOMP)*, 2003.

[59] P. Peris-Lopez, J. C. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda, "LMAP: A Real Lightweight Mutual Authentication Protocol for Low-cost RFID tags," *Workshop on RFID Security (RFIDSec)*, 2006.

[60] L. Bolotnyy and G. Robins, "Physically Unclonable Function-Based Security and Privacy in RFID Systems," *Pervasive Computing and Communications, 2007. PerCom '07. Fifth Annual IEEE International Conference on*, pp. 211–220, March 2007.

[61] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," *Proceedings of the ACM Conference on Computer and Communications Security*, 2002.

[62] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," *Proceedings of the Computer Security Applications Conference*, pp. 149–160, 2002.

[63] D. Lim, J. Lee, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 13, pp. 1200–1205, October 2005.

[64] K. Yuksel, "Universal Hashing for Ultra-Low-Power Cryptographic Hardware Applications," Tech. Rep., Worcester Polytechnic Institute, 2004.

[65] P.-K. Leung, C.-S. Choy, C.-F. Chan, and K.-P. Pun, "An optimal normal basis elliptic curve cryptoprocessor for inductive RFID application," *Proceedings IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 309–312, 2006.

[66] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," *Security in Pervasive Computing*, Vol. 2802 of *Lecture Notes in Computer Science*, pp. 201–212, 2004.

[67] S. A. Weis, "Security and Privacy in Radio-Frequency Identification Devices," Master's thesis, Massachusetts Institute of Technology, 2003.

[68] Y. Cui, K. Kobara, K. Matsuura, and H. Imai, "Lightweight Asymmetric Privacy-Preserving Authentication Protocols Secure against Active Attack," *Proc. of the Pervasive Computing and Communications Workshop*, pp. 223–228, 2007.

[69] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Problems of Control Information Theory*, Vol. 15, No. 2, pp. 159–166, 1986.

[70] I. Verbauwhede, P. Schaumont, and H. Kuo, "Design and Performance Testing of a 2.29Gb/s Rijndael Processor," *IEEE Journal of Solid-State Circuits*, Vol. 38, pp. 569–572, March 2003.

[71] D. J. Wheeler and R. M. Needham, "TEA, a tiny encryption algorithm," *Proc. of the Workshop Fast Software Encryption*, Vol. 1008 of *Lecture Notes in Computer Science*, pp. 363–366, 1994.

[72] P. Israsena, "Securing ubiquitous and low-cost RFID using tiny encryption algorithm," *International Symposium on Wireless Pervasive Computing*, 2006.

[73] M. Kim, J. Ryou, Y. Choi, and S. Jun, "Low-cost Cryptographic Circuits for Authentication in Radio Frequency Identification Systems," *IEEE International Symposium on Consumer Electronics (ISCE)*, pp. 1–5, 2006.

[74] W. Stallings, *Cryptography and Network Security*. Pearson Prentice Hall, 2006.

[75] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede, "An Elliptic Curve Processor Suitable For RFID-Tags." Cryptology ePrint Archive, Report 2006/227, 2006.

[76] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede, "Public-Key Cryptography for RFID-Tags," *Proc. of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW)*, pp. 217–222, 2007.

[77] A. Juels, "Strengthening EPC tags against cloning," *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*, (New York, NY, USA), pp. 67–76, ACM Press, 2005.

[78] International Standards Organization, "ISO/IEC 14443-4:2001." Standard Specification, 2001.

[79] International Standards Organization, "ISO/IEC 7816-3:2006." Standard Specification, 2006.

[80] K. Finkenzeller, *RFID Handbook: Radio-frequency identification fundamentals and applications*. Wiley, 1999.

[81] NXP/Philips, *MiFare DESfire: Contactless Multi-Application IC with DES and 3DES Security MF3 IC D40*, 3 ed., 2002.

[82] D. Coppersmith, "The data encryption standard (DES) and its strength against attacks," *IBM Journal of Research and Development*, Vol. 30, No. 3, pp. 243–250, 1994.

[83] A. Juels, D. Molnar, and D. Wagner, "Security and Privacy Issues in E-passports," *Proc. of Security and Privacy for Emerging Areas in Communications Networks*, pp. 74–88, 2005.

[84] International Civil Aviation Organization, *PIK for Machine Readable Travel Documents offering ICC Read-Only Access*, 1.1 ed., October 2004.

[85] International Standards Organization, "ISO/IEC FDIS 18185-1:2006." Standard Specification, 2006. under development.

[86] A. K. Jones, R. Hoare, S. Dontharaju, S. Tung, R. Sprang, J. Fazekas, J. T. Cain, and M. H. Mickle, "An Automated, FPGA-based Reconfigurable, Low-power RFID Tag," *Proceedings of the 43rd Design Automation Conference (DAC)*, pp. 131–136, ACM, July 2006.

[87] P. J. Hawrylak, L. Mats, J. T. Cain, A. K. Jones, S. Tung, and M. H. Mickle, "Ultra Low-power Computing Systems for Wireless Devices," *International Review on Computers and Software (IRECOS)*, Vol. 1, No. 1, pp. 1–10, 2006.

[88] A. K. Jones, S. Dontharaju, S. Tung, L. Mats, P. Hawrylak, R. Hoare, J. T. Cain, and M. H. Mickle, "Radio Frequency Identification Prototyping," *ACM Transactions on Design Automation for Electronic Systems (TODAES)*, 2006. in review since September 2006.

[89] A. K. Jones, R. Hoare, D. Kusic, G. Mehta, J. Fazekas, and J. Foster, "Reducing Power while Increasing Performance with SuperCISC," *ACM Transactions on Embedded Computing Systems (TECS)*, Vol. 5, pp. 1–29, August 2006.

[90] O. K. Incorporatd, "Opal Kelly XEM3001v2 User's Manual." http://www.opalkelly.com, 2005.

[91] O. K. Incorporatd, "Opal Kelly FrontPanel User's Manaul." http://www.opalkelly.com, 2005.

[92] EPCglobal., "Class 1 Generation 2 UHF Air Interface Protocol Standard Version 1.0.9." W3C Recommendation, 1998. http://www.epcglobalinc.org/standards/.

[93] M. Roberti, "SAMSys to Unveil RFID Reader Platform," *RFID Journal*, 2006.

[94] M. C. O'Conner, "Sirit to Buy SAMSys Technologies," *RFID Journal*, 2006.

[95] S. Dontharaju, S. Tung, J. T. Cain, L. Mats, M. H. Mickle, and A. K. Jones, "A Design Automation and Power Estimation Flow for RFID Systems," *ACM Transactions on Design Automation for Electronic Systems (TODAES)*, in review since Sept. 2007.

[96] D. Engels and S. Sarma, "The Reader Collision Problem," Nov. 2001. White paper MITAUTOID-WH-007, Auto-ID Center.

[97] R.Want, "The Magic of RFID," Oct. 2004. ACM Queue.

[98] S. A. Delichatsios, D. W. Engels, L. Ukkonen, and L. Sydanheimo, "Albano multidimensional UHF passive RFID tag antenna designs," *International Journal of Radio Frequency Identification Technology and Application*, Vol. 1, No. 1, pp. 24 – 40, 2006.

[99] L. Ukkonen, M. Schaffrath, J. Kataja, L. Sydanheimo, and M. Kivikoski, "Evolutionary RFID tag antenna design for paper industry applications," *International Journal of Radio Frequency Identification Technology and Application*, Vol. 1, No. 1, pp. 107 – 122, 2006.

[100] J. D. Porter, R. E. Billo, and M. H. Mickle, "Effect of active interference on the performance of radio frequency identification systems," *International Journal of Radio Frequency Identification Technology and Application*, Vol. 1, No. 1, pp. 4 – 23, 2006.

[101] K. Penttila, L. Sydanheimo, and M. Kivikoski, "Implementation of Tx/Rx isolation in an RFID reader," *International Journal of Radio Frequency Identification Technology and Application*, Vol. 1, No. 1, pp. 74 – 89, 2006.

[102] S. R. Dontharaju, *Design Automation for Low Power RFID Tags*. PhD thesis, University of Pittsburgh, 2007.

[103] IEEE Computer Society, "IEEE Standard 802.3-2005," December 2005.

[104] American National Standards Institute, "ANSI/TIA/EIA-422-B." Standard Specification, May 1994.

[105] ATIS Committee T1A1, "ATIS Telecom Glossary 2000," Tech. Rep. T1.523-2001, Alliance for Telecommunications Industry Solutions (ATIS), 2001.

[106] IEEE Computer Society, "IEEE Standard 802.5-1998E," May 1998.

[107] D. M. Levis, B. W. Thomson, P. I. P. Boulton, and E. S. Lee, "Transforming Bit-Serial Communication Circuits into Fast Parallel VLSI Implementations," *IEEE Journal of Solid-State Circuits*, Vol. 23, pp. 549–557, April 1988.

[108] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC): Stengths and Weaknesses," *Proc. of the Workshop on RFID Security*, 2006.