Implementation of a Low-Resource Authentication Device

Hannes GROß
hannes.gross@student.tugraz.at

Institute for Applied Information
Processing and Communications (IAIK)
Graz University of Technology
Inffeldgasse 16a
8010 Graz, Austria

Master's Thesis

Supervisors: Martin Feldhofer and Thomas Plos
Assessor: Karl-Christian Posch

February, 2013

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …………………………… ………………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…………………………… ………………………………………………..
date (signature)

Acknowledgements

I wish to thank everyone who helped me complete this thesis. I would like to thank Martin Feldhofer for being a great supervisor and for giving me the opportunity to write my thesis on the basis of a practical work for NXP Semiconductors. Special thanks goes out to Thomas Plos for supporting all of my works during my study and for always taking his time to help me with advice and encouraging words. Furthermore, I want to thank the rest of the members and former members of the Institute for Applied Information Processing and Communications (IAIK) for giving me the chance to work in a great team: Manfred Aigner, Sandra Dominikus, Michael Hutter, Mario Kirschbaum and Jörn-Marc Schmidt. Finally, I want to thank my loved girlfriend, my family and friends for their moral support.

Abstract

The development of semiconductor technology with increasingly smaller structure sizes — that are beyond the wavelength of visible light — and more energy efficient transistors also led to a further development of the Radio-Frequency Identification technology (abbreviated RFID). The functionality of passive (without an additional voltage source) RFID tags increased from the transmission of identification numbers to complex communication protocols with dynamic memory management. In order to cope with the security and privacy requirements of such systems the usage of powerful cryptographic algorithms is necessary. This thesis discusses the integration of a standardized cryptographic algorithm into a digital tag design and the implementation of cryptographic authentication methods from the perspective of a digital designer. Furthermore, modern hardware verification methods for the functional verification of complex digital systems are shown.

Keywords: Radio-Frequency Identification (RFID), authentication, privacy, hardware verification, Advanced Encryption Standard (AES), EPC Gen 2 standard

Kurzfassung

Durch die Weiterentwicklung der Halbleitertechnologie hin zu immer kleineren Strukturgrößen (jenseits der Wellenlänge des sichtbaren Lichts), mit immer energieeffizienteren Transistoren, entwickelte sich auch Radiofrequenzidentifikation (abgekürzt RFID) stetig weiter. Die Funktionalität von passiven (ohne zusätzliche Spannungsquelle versorgten) RFID-Tags entwickelte sich vom Übertragen von Identifikationsnummern, hin zu komplexen Kommunikationsprotokollen mit dynamischer Speicherverwaltung. Um den Anforderungen in Bezug auf Sicherheit und Privatsphäre, die an solche Systeme gestellt werden, gerecht zu werden, ist der Einsatz von leistungsfähigen kryptografischen Algorithmen unabdingbar geworden. In der vorliegenden Arbeit wird die Integration eines standardisierten kryptografischen Algorithmus in ein digitales Tag-Design, sowie die Implementierung von kryptografischen Authentifizierungsmethoden, aus Sicht eines Digitaldesigners behandelt. Weiters werden moderne Methoden zur funktionalen Verifikation komplexer Digitalsysteme vorgestellt.

Stichwörter: Radiofrequenzidentifikation (RFID), Authentifizierung, Privatsphäre, Hardwareverifikation, Advanced Encryption Standard (AES), EPC Gen 2 Standard

Contents

List of Figures

List of Tables

Chapter 1

Introduction - A Brief Historical Overview of RFID

Radio waves—respectively electromagnetic waves—are neither an invention of the modern age nor a human invention. The first natural electromagnetic wave sources are almost as old as the universe itself (13.7 billion years). Even today the residuals of the Big Bang—which is the most widely accepted theory about the origin of the universe (see Berhorst [6], page 27)—are still measurable as the so-called cosmological background radiation. Furthermore, there are other natural sources of radio waves like cosmological pulsars or radio galaxies. The detection of radio waves and the classification as a part of the electromagnetic spectrum, however, is part of the history of the modern age. See Figure 1.1 for an illustrated historical overview of the developments that led to today's RFID technology. All dates in this text that lack an explicit citation were taken from Jeremy Landt's article "The history of RFID" [40].
Around 1800 Friedrich Wilhelm Herschel was the first one to find out that the light contains more than just the small range that is visible to the human eye. Herschel used a prism to split the sunlight into its wave spectrum and detected the infrared radiation above the end of the visible red light. Faraday explored the influence of magnetism on light by sending linearly polarized light through a magnetic field in 1846. The discovered effect was a rotation of the polarization plane of the light. He concluded that light and magnetism are related and that light could be produced by transversal oscillation of the electromagnetic lines of force. Faraday's ideas (et al.) led to the well known Maxwell's equations which were propagated by James Clerk Maxwell in 1864, and describe the behavior of electric and magnetic wave fields and their interaction with matter. The following citation from Maxwell (taken from [3]) shows that he was the first to recognize that light—in its visible and invisible form—is an electromagnetic wave.

> "This velocity is so nearly that of light that it seems we have strong reason to conclude that light itself (including radiant heat and other radiations) is an electromagnetic disturbance in the form of waves propagated through the electromagnetic field according to electromagnetic laws."

In 1887 Heinrich Hertz developed an experiment to proof the existence of electromagnetic waves. The device he used was called the "Hertzian Oscillator" and consisted of a transmitter and a receiver (open metal ring). On the transmitter side there was a construction that generated a strong electric spark. This electric spark also caused an electromagnetic wave and initiated an electric current on the receiver's dipole antenna. Furthermore,

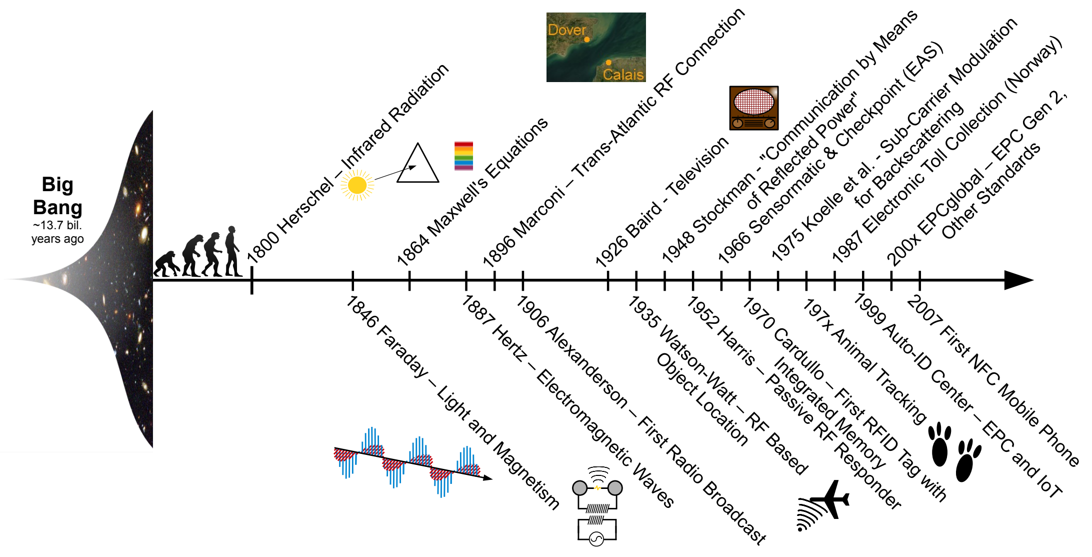

Big Bang
~13.7 bil. years ago
1800 Herschel – Infrared Radiation
1846 Faraday – Light and Magnetism
1864 Maxwell's Equations
1887 Hertz – Electromagnetic Waves
Dover
Calais
1896 Marconi – Trans-Atlantic RF Connection
1906 Alexanderson – First Radio Broadcast
1926 Baird - Television
1935 Watson-Watt – RF Based Object Location
1948 Stockman - "Communication by Means of Reflected Power"
1952 Harris – Passive RF Responder
1966 Sensormatic & Checkpoint (EAS)
1970 Cardullo – First RFID Tag with Integrated Memory
1975 Koelle et al. - Sub-Carrier Modulation for Backscattering
197x Animal Tracking
1987 Electronic Toll Collection (Norway)
1999 Auto-ID Center – EPC and IoT
200x EPCglobal – EPC Gen 2, Other Standards
2007 First NFC Mobile Phone

Figure 1.1: Historical Overview of the Evolution of RF(ID) Technology

Hertz calculated the speed of the electromagnetic wave and confirmed Maxwell's thesis that it equals the speed of light. Guglielmo Marconi shall be deemed to be the pioneer of wireless communication. He studied Hertz's work and made experiments with the intention to transmit and receive information. After experiments with a shorter distance of receiver and transmitter he established a radio connection across the English Channel in 1896. Ten years later (1906) the technical progress evolved so far that the first radio broadcast could be set up by E. F. W. Alexanderson. Alexanderson developed a radio transmitter—the so-called Alexanderson alternator—which was powerful enough to transmit radio waves from Brant Rock, Massachusetts to the Caribbean Sea. In 1926 John L. Baird [35] patented a radio transmitter for object detection and about one year later a construction to receive and display this information was patented. The television set was developed and by the beginning of the 1930s the British Broadcasting Company began to send the first television pictures using a method that was developed by Baird. The Scottish physicist Robert Watson-Watt did research on the reflexion of radio waves in the atmosphere and realized that this issue could be used for the location of objects. The first airplane was located [58] using a refinement of the method that had been patented by Watson-Watt in 1935. Harry Stockman studied "Communication by Means of Reflected Power" and wrote an interesting article [54] in 1948. In this article he discussed a communication form that is nowadays commonly used in ultra-high frequency and microwave RFID technology and is called "backscattering". The communication principle is that the amount of power which is reflected by the other side of the communication path is varied according to a communication protocol in order to transmit information. F. L. Vernon showed in an article released in 1952 the application of "homodyne detection" for microwave-based communication (see [57]) and stated:

"The measurement of phase at microwave frequencies is important in the development of microwave antenna components and in the measurement of impedance properties of microwave components in general. [. . .] An analysis of the characteristics of homodyne detection shows that they are suitable for application in many ways at microwave frequencies."

At about the same time, D. B. Harris worked on "Radio Transmission Systems with Modulatable Passive Responder". In this patent Harris describes five ways to establish a radio connection with a passive (without external power supply) responder. One of his proposed solutions is shown in Figure 1.2. The figure contains a "centrally located station" (analog to an RFID reader) that consists of an amplitude-modulation transmitter (1) and a receiver (2). When the transmitter sends the signal —which is connected trough the line (4) over the hybrid coil (3) to the transmitter—over the transmission antenna, a part of the transmitted signal is fed back over the phase network (15) in opposite phase. This needs to be done to compensate the signals that are received by the antenna (6) and were caused by the transmitter on the "centrally located station" side. The signals from the "remotely located station" are not touched by the phase network and are received trough the line (4). In analogy to a passive UHF RFID tag front-end, this device consists of a dipole antenna (7) for receiving and sending signals. The tuning network consist of the capacitors (8) and (10). The transformer (9) is the link between the antenna and the modulator/demodulator network (11)—and vice versa. After the demodulation stage the signal is low-pass filtered by the capacitor (12) and sent to the telephone device (13). For the other communication direction there exists a microphone (14).
In 1966 the first companies Sensormatic and Checkpoint appeared with the focus on electronic article surveillance (EAS). This technique protects stores against shop lifting by attaching transponders on consumer goods like clothing, books, etcetera. When a product leaves the store without deactivating the transponder, the alarm of the surveillance system is triggered. The technique for EAS products is simple, inexpensive and effective against shop lifting. The EAS labels used either microwave or inductive technology to generate a response signal from a reader field. Moreover, it was the first commercial RFID product. Mario W. Cardullo et al. designed and constructed the first prototype of a modern RFID tag around 1970. Figure 1.3 shows a block diagram of a passive transponder device which contains a random-access memory. The memory unit is accessible through commands that are received over the radio field and decoded by the internal logic. At the same time the radio field loads a capacitor that powers the transponder device. The stated field of application was automatic payment of the toll roads for automobiles. Besides the radio wave realization of the transponder, the patent also comprises the use of light and sound waves. RFID became an interesting subject area and many companies, universities, research laboratories and others worked actively to develop new RFID technology. One interesting work at this time, "Short-Range Radio-Telemetry for Electronic Identification, Using Modulated Backscatter," was written by A. Koelle, S. Depp and R. Freyman from the Los Alamos Scientific Laboratory, New Mexico [38] in 1975. This work showed the use of a (20 kHz) sub-carrier that is modulated on the radio-frequency signal of the backscattered answer by varying the load of the transponder antenna. This approach made it possible to avoid problems that occurred when homodyne detection or other modulated backscattering methods were used. The following citation was taken from the article.

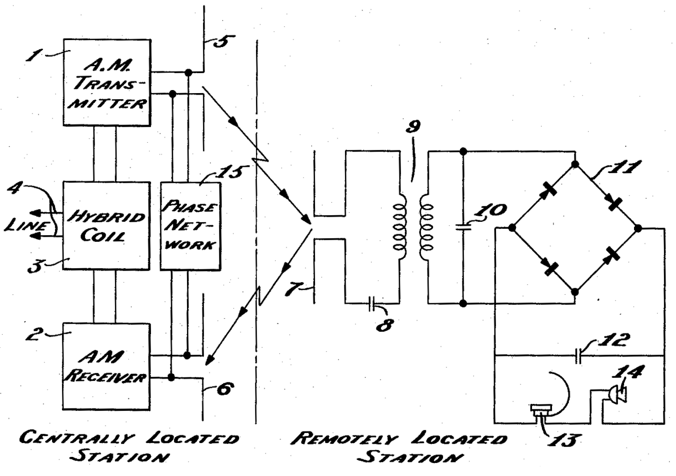


Figure 1.2: Figure from D.B. Harris patent [26] for "Radio Transmission Systems with Modulatable Passive Responder" from 1952

"The problem of how to modulate the backscatter is solved in a way that is both effective and economical, by varying the load on the transponder antenna rectifier. We believe that these innovations make the modulated backscatter method practicable and that the economy and simplicity of the method will encourage wide use in electronic ID systems and other short-range telemetry applications."

The development of RFID technology shows that Koelle, Depp and Freyman were right with their prediction. It was the beginning of the practical use of passive RFID tags. Also companies began to develop RFID technology at this time. At the end of the 1970s it was used for animal tracking for the first time. In the 1980s, the low-power CMOS technology and the development of personal computers became an enabler for further distribution of RFID systems. Personal computers allowed to establish the automated collection of data in an economic way. This led to the first electronic toll collection for automobiles in Norway in 1987 with followers in the USA. Four years later, the Association of American Railroads (AAR) adopted an RFID system that was developed by J. Landt as industry standard for tracking the train positions and their status. Another three years later every train in the USA was equipped with RFID technology. The 1990s were characterized by commercial use of RFID products. Tolling and train tracking was used in many countries worldwide and a European C.E.N standard for tolling was developed. Other RFID applications like ski passes, car access and contactless payment were introduced, e.g., the Texas Instruments Registration and Identification System (abbreviated TIRIS) to control the start of the car engine. Moreover, the research on RFID continued and new research facilities were founded like the Auto-ID Center in 1999 (Finkenzeller [15], page 309) which was a non-profit collaboration between academic institutions and companies. The goal of this institution was to develop an infrastructure (cf. Internet of Things) to track goods worldwide that carry Electronic Product Codes (abbreviated EPC). At the same time the first books appeared with the focus on all aspects of RFID technology. Finkenzeller's "RFID Handbook: Fundamentals and Applications..." [16] is probably the most popular example.

In 2003 the EPCglobal Inc. was founded and it continued the standardization work of

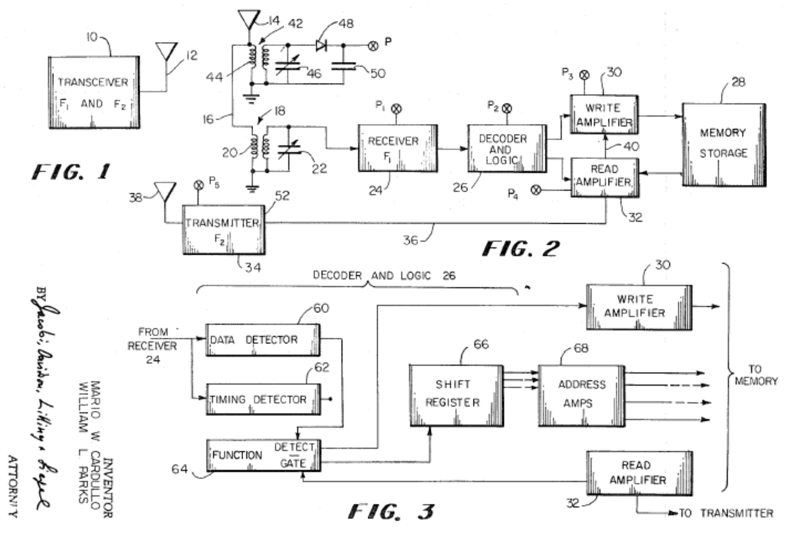


Figure 1.3: Figure from Mario Cardullo's et al. patent [9] of "Transponder Apparatus and System" from 1973

the Auto-ID Center. EPCglobal developed the EPC Gen 2 standard which defined new communication standards between transponders and readers. The EPC Gen 2 standard was more suitable to become the international standard for electronic product codes than its predecessor since it solved some weaknesses of the first EPC standard (Finkenzeller [15], page 309). Besides the EPC Gen 2 standard other standards were developed around 2000 such as ISO/IEC 14443 (2000 - 2008 [30]) for proximity cards, ISO/IEC 15693 (2000 - 2010 [30]) for vicinity coupling cards or ISO/IEC 18000 (2004 - 2008 [30] for item identification in logistics. The first mobile phone with near-field communication (abbreviated NFC, ISO/IEC 14443 based) support was released around 2007 by Nokia. Over the years the academic interest in RFID technology increased which can be seen in Figure 1.4 showing the number of search results on IEEE Xplore [28] for the keyword "RFID" according to the year of the publication from 1991 to 2012. From 2001 to 2007 the number of publications doubled almost every year from 10 publications to 1068. The maximum was reached in 2010 with 1590 RFID-related publications. The increased number of publications is explicable with a better academic infrastructure like conferences on RFID topics and international working groups. Also the enhanced computing power of tags led to an increased complexity of tag functionality which was an opener for new research fields such as security or privacy for RFID systems, smart cards, the Internet of Things, etcetera.
I want to conclude this brief overview of the history of RFID with a quote from Jeremy Landt's article "The history of RFID" [40] which was published in 2005.

> "We have a great many developments to look forward to, history continues to teach us that."

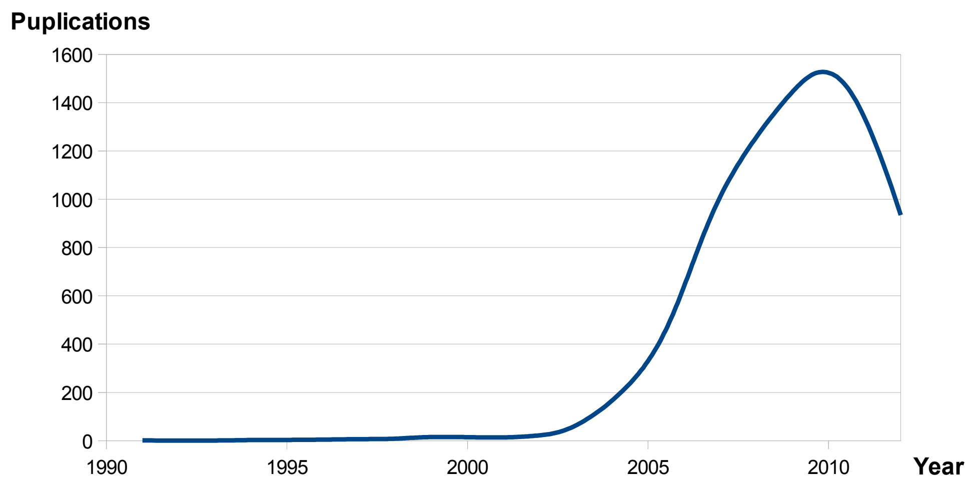


Figure 1.4: Development of Academic Interest in RFID, Demonstrated on IEEE Xplore Search Results

The remainder of this work is structured as follows:
Chapter 2 gives an overview of state-of-the-art RFID technology and explains the underlying principles on different design layers. A subsection shows the physics behind electromagnetic waves and how they can be used to transport both information and energy. The chapter concludes with a detailed introduction of the EPC Gen 2 communication protocol.
Chapter 3 concentrates on authentication methods and especially on challenge-response protocols that are based on symmetric-key cryptography. Then the Advanced Encryption Standard (AES) is discussed as a popular example of a symmetric-key block cipher. The rest of the chapter focuses on privacy for RFID tags.
Chapter 4 is about a very important but still underrated topic in hardware design: The systematic verification of an implemented system based on the system specification. A typical hardware design flow from the specification on system level to the fabrication process is explained. Then different ways measuring the progress of a verification process—like code coverage or functional coverage—are compared. The main part of this chapter is about the comparison of classical verification methods like a Verilog or Tcl test bench to modern design methodologies that can be used with SystemVerilog or SystemC. The remaining part of the chapter is about the executable specification that was used for the practical part of this master's work and the post-silicon validation of the chip.
Chapter 5 focuses on the practical part of the work and begins with a system overview of the digital part of the chip. Besides the description of the system components also the implemented authentication methods are explained. During the implementation process some problems occurred which are related to the asynchronous parts of the chip design. These implementation-related problems and their solutions are discussed in an own section.
Chapter 6 summarizes this work and draws conclusions.

Chapter 2

State-of-the-Art RFID Technology

The following section provides a short introduction to RFID technology like it is used today. In the first section the different kinds of RFID tags and their differentiation features are explained, like frequencies, coupling types, active or passive tags, etcetera. Section 2.2 is about the physical and theoretical background of radio-frequency communication with special regard to electromagnetic waves which are important for UHF communication. Finally, Section 2.3 discusses the high-level functionality of the EPC Gen 2 protocol by exploring the different protocol phases. The section starts with the selection of population subsets of tags and ends with the access of individual tags.

2.1 Basic Principles of RFID

The basic RFID-related terms and differentiation features can be explained best on the basis of the different abstraction layers in the design process of a transponder device (see Figure 2.1). On the top-level of the abstraction layer—seen from a top-down view—there is the **Application Layer**, which of course influences all underlying layers. The figure shows some examples for the application layer which result in different requirements and constraints for the design. An electronic article surveillance system (EAS) results in a passively driven low-cost transponder design with high constraints to range and power supply and low requirements for security—if needed at all (cf. light-weight block ciphers versus asymmetric cryptography). On the other hand, a mobile phone with near-field communication functionality does not need to be as power constrained as an EAS tag but has higher requirements for security and privacy. Also the way of implementing the transponders is totally different, since an EAS transponder is mostly analog electronic and an NFC component for a mobile phone might have its own microcontroller or is controlled by the operating system of the mobile phone (hardware/software co-design).
The **System Layer** distinguishes between the different kinds of transponders (1-bit transponder, tags, smartcards) and the implementation of their internal control logic. The type of available memory is also a characteristic of the transponder design because the availability of a non-volatile memory allows the configuration of the transponders or to change their firmware. According to the application area, the transponders can come in different formats also-called housings, e.g., the transponders for animal tracking use a glass housing which is implanted under the skin. For logistic systems the transponder needs to be invulnerable against physical transformation and is part of a thin label that can be attached onto a surface.

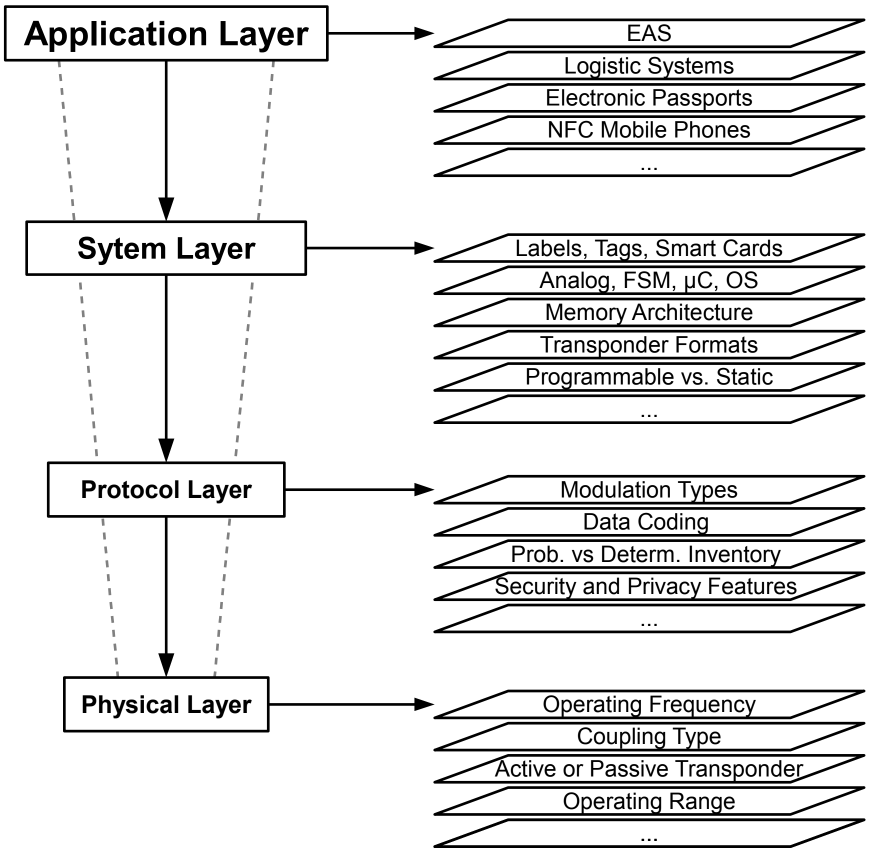


Figure 2.1: Differentiation Features of RFID Systems on Different Layers

On the **Protocol Layer** the differentiation features are basically defined by the actual protocol that is implemented which defines the reader-to-transponder communication and vice versa. This implies the different modulation types like ASK, FSK, PSK, the data encoding (NRZ, Manchester coding, Miller coding, etcetera) and the data and frame formats. The inventory strategy is also part of the protocol and can either be deterministic or probabilistic (cf. EPC Gen 2 inventory). The protocol also defines some kind of data-integrity feature for data frames which is at most a cyclic redundancy check (abbreviated CRC) or some kind of parity checking. Standardized security and privacy is a rather new topic for RFID and the standardization of cryptographic suits is still in progress. Product-related security like it is used for car access or ticketing systems for public transportation have already been used before. The security protocols for these applications were mostly designed inside one company without publication of the protocol. This type of security provisioning is called “security by obscurity” and is treated to be bad practice with high consensus of security experts. This led to many hacked systems in the past. One example for this practice is the block cipher “KeeLoq” which is used for remote keyless entry (cars) or passive entry systems. In 2007, Bogdanov [7] presented attacks and weaknesses of the cipher.

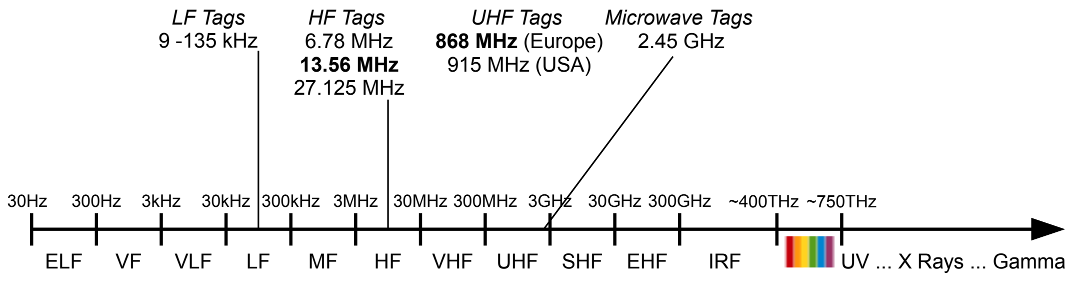


Figure 2.2: Electromagnetic Frequency Spectrum with Frequency Ranges, Names and Relevant RFID Frequencies

The differentiation on the **Physical Layer** is strongly related to the type of the RFID transponder respectively its operating frequency, e.g., microwave, HF, UHF, etcetera. This also defines the way the communication channel is implemented (coupling type) and the method that is used to transmit and receive information (e.g., backscattering or load modulation). The transponders can either be “active” which means that they have an own or additional power supply or “passive” which means that the whole operating energy is extracted from the reader field. The operating range is also related to the operating frequency but varies with the actual implementation of the tag.
The following subsections are named after the described abstraction layers and give more detailed information on the related differentiation features.

2.1.1 Physical-Layer Differentiation

Operating Frequency: According to Finkenzeller (see [16], page 154) and Miller (see [43], page 3 and 4) the spectrum of electromagnetic waves consists of the frequency ranges shown in Figure 2.2. The spectrum also contains the radio wave ranges which are important for RFID applications. ISM frequencies—which stands for “Industrial Scientific and Medical”—are internationally reserved for the use of high-frequency devices. This does not only imply radio frequency transmission devices but also microwave ovens for example. The most important ISM frequencies for RFID transponders are 13.56 MHz for HF tags, 868 MHz for UHF tags and 2.45 GHz for microwave tags. Short-range device frequencies (abbreviated SRD) are public usable frequencies in the range between 865 MHz and 868 MHz. An application in this range does not need to be authorized or registered (in Europe) which is a big advantage for the use in RFID in terms of costs. Another part of the radio spectrum that can be used for RFID purposes is the range between 9 and 135 kHz which is used for example for sensor devices that are placed inside the rumen of cattle (see [16], page 26).

Coupling: A coupling type that can be used for LF and HF transponder ranges is the **inductive coupling**, where the medium used for data transfer and for power supply is an alternating magnetic field (Figure 2.3). The field is provided by the resonate circuit of the reader and induces an alternating current in the transponder coil that is tuned to the operating frequency of the reader with a tuning capacitor (C_{tune}). Another capacitor (C_{pow}) is used for the passive supply of the transponders and serves as a stabilized

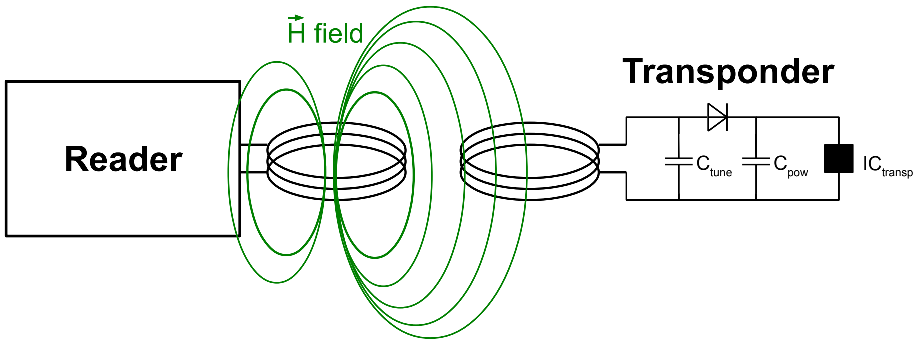


Figure 2.3: Inductively Coupled Reader and Transponder

power supply for the integrated circuit. This form of coupling works effectively only in the near field of the reader coil (distance $< \lambda/2\pi$, e.g., for 13.56 MHz $\sim$ 3.5 m) for the transmission of data from the transponder to the reader. The distance for passively driven tags is much smaller than this definition of the near field. When the distance between reader and transponder exceeds the near field and enters the far field, the transmission of data by **electromagnetic coupling** is more effective. For this type of coupling the electromagnetic waves are used instead of the magnetic $\vec{H}$ field which changes the design of the transponder antennas from coils to dipoles. Figure 5.13 in Section 5.1.7 shows a frontend of a UHF tag and explains the data processing and power supply inside the passive transponder. The communication principle is the same as for radar technology. Depending on the emitted wavelength of the reader and the size of the antenna (or object for radar) a certain amount of power is reflected. The maximum amount of power is reached when the antenna circuit of the transponder is in resonance with the electromagnetic field emitted by the reader. Backscattering works by changing the load that is connected to the antenna of the transponder which changes the resonance frequency of the resonate circuit respectively the amplitude of the reflected power. This principle can be used to transfer data from the transponder to the reader. Other coupling types are **close coupling**, which is suitable for transponders that need high amounts of energy for passive operation for distances between 0.1 and 1 cm, and **electrical coupling** where the reader and the tag are coupled capacitively. The latter two coupling types have only small relevance for RFID tags.

Power Supply: The power for transponders can be supplied either with or without an additional power supply. Transponders that do not need an additional power source are called **passive** transponders which is the same nomenclature as for RFID tags. Active radio frequency transponders have a power supply and are able to produce their own high-frequency field that is independent from the reader field. RFID tags are not able to produce their own high-frequency field even if they have an additional power supply. In terms of radio transponders RFID tags with a battery supply are therefore called **semi-active** or **semi-passive**. For the classification of RFID tags it is also common to call these tags **active**.

LF *135 kHz*	**HF** *13.56 MHz*	**UHF** *868 MHz*		**Microwave** *2.45 GHz*	
passive	passive	passive	active	passive	active
0.5 m	1.5 m	3-5 m	15 m	< 1 m	15 m

Table 2.1: Comparison of Maximum Operating Ranges for RFID Tags

Range: The maximum distance between reader and responder given in Table 2.1 are typical values and are supposed to show the tendency of how the range is influenced by the operating frequency, coupling and the power supply of the tag. The actual possible range also depends on the strength of the reader field, the use of a directional radio (UHF and microwave only), the tag design (power consumption), the environment where the tags are used, etcetera. For more information see Fleisch et al. [17] pages 78 and 79, Finkenzeller [16] pages 21 - 26 and 46 or Ward et al. [42] on page 9.

2.1.2 Protocol-Layer Differentiation

Data Coding: Data coding defines how binary values are represented in the baseband signal. This can either be by **direct coding** of the actual baseband signal value (NRZ coding or unipolar RZ coding) or a **transitional coding** of the baseband signal (Manchester coding). Other codings rely on the baseband signal value of the last symbol that was sent. Examples for this coding style are Differential code for direct coding of the baseband signal and DBP or Miller coding for transitional coding. The reasons for using a particular coding style are the signal spectrum after the modulation, the continuity of the power supply for passive transponders and the recognizability of transmission errors (bad symbols) or collisions, e.g., caused by two tags that are sending data simultaneously. Figure 2.4 shows examples for different coding types. To demonstrate the different behavior of the codings Figure 2.5 shows a communication with two involved tags and one reader. The tags are sending data simultaneously, on the left side with Differential coding and on the right side with Manchester coding. The first bits of both tags are the same (logic '1') therefore no collision occurs. In the second bit slot *Tag 1* sends another '1' and *Tag 2* a '0'. With Differential coding the reader receives a "correct" symbol without detecting a collision, while the reader on the right side receives a bad symbol for the Manchester coded data and notices that a collision occurred. The third received bit for the Differential coded data is a '0' which does not occur in either of the tag signals. In a worst-case scenario the reader on the left receives a correct frame or command with inconsistent data without recognizing that something went wrong. Usually some kind of consistency check data is sent like parity or a CRC checksum but there is still the chance that multiple errors are not detected. The disadvantages for Manchester coding are a higher sampling rate, which is necessary to decode the same amount of data as for Differential coding, and a more complex encoding and decoding logic.

Modulation Types: Modulation is the way of influencing carrier-signal parameters (amplitude, frequency or phase) according to the encoded baseband signal and the modulation type so that the receiver is able to extract the data from the carrier signal. The frequency of the signal that is used as a carrier signal to transmit the coded data is much higher compared to the encoded data signal, independent from the actual coupling type (e.g.,

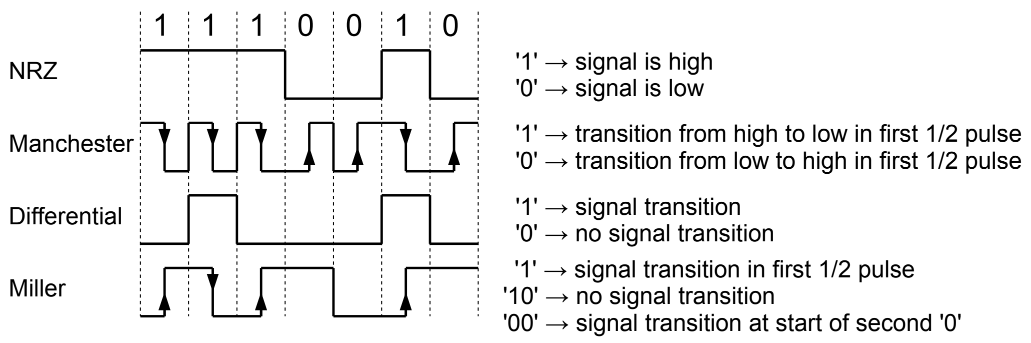

1 1 1 0 0 1 0
NRZ
Manchester
Differential
Miller
'1' → signal is high
'0' → signal is low
'1' → transition from high to low in first 1/2 pulse
'0' → transition from low to high in first 1/2 pulse
'1' → signal transition
'0' → no signal transition
'1' → signal transition in first 1/2 pulse
'10' → no signal transition
'00' → signal transition at start of second '0'

Figure 2.4: Coding Examples for Direct-, Transitional-, Direct Differential-, and Transitional Differential Codings

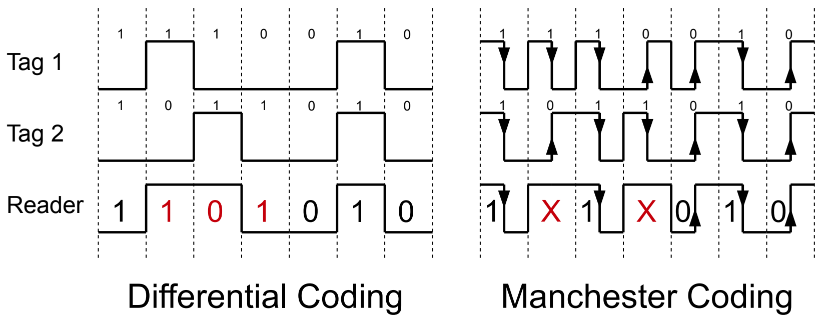


Figure 2.5: Two Tags Transmitting Data Simultaneously to a Reader with Two Different Codings

electromagnetic waves or inductive coupling).
The simplest form of modulation is the *Amplitude-Shift Keying* (abbreviated ASK) with a modulation factor of 100% which is also-called *on-off keying* because the carrier signal is turned on or off. One disadvantage becomes evident when a coding type like NRZ is used and a long trail of zeros should be transmitted to a passive tag. Since the field is turned off during the whole time while zeros are transmitted but the tag still consumes power the tag will sooner or later run out of energy. This effect can be avoided or reduced when a modulation index smaller then 100% is used. Figure 2.6 shows an example of ASK modulation with a modulation index "m" of 100% and 50%. The modulation can be written as a multiplication of the carrier signal with the weighted binary-coded data or baseband signal (Equation 2.1).

$$u_{ASK}(t) = u_{carrier}(t) - m * (1 - u_{signal}(t)) * u_{carrier}(t) \tag{2.1}$$

Frequency-Shift Keying (abbreviated FSK) uses a change of the frequency component of the carrier signal to transmit information. The most common implementation uses two

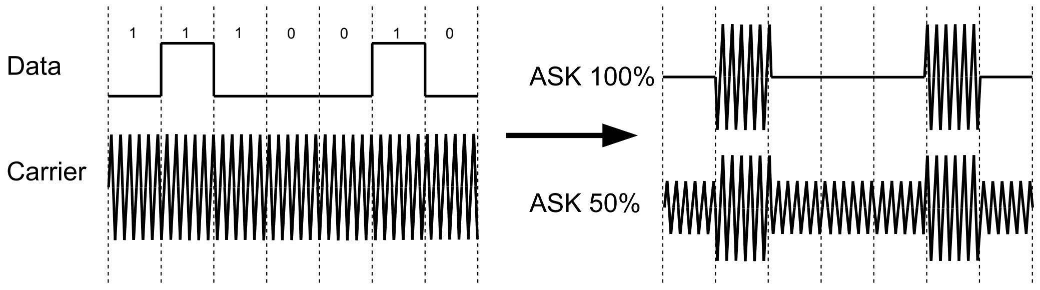


Figure 2.6: ASK Modulation Scheme with a Differential Coded Data Signal, the Carrier Signal and Different Modulation Factors

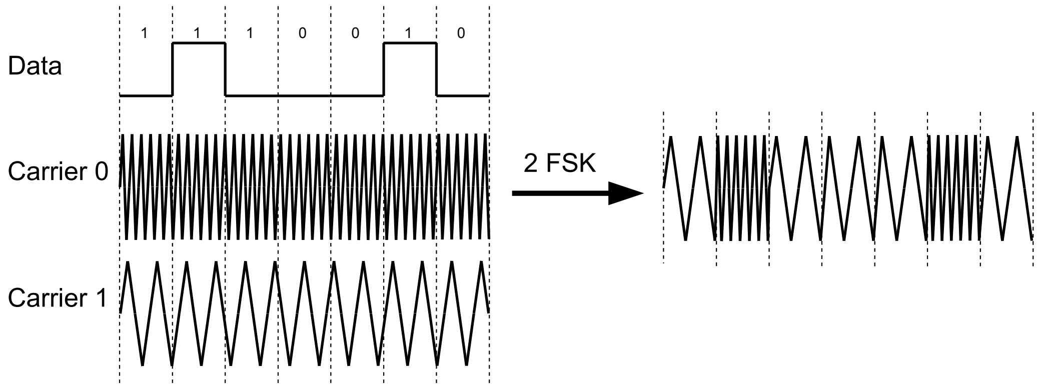


Figure 2.7: 2 FSK Modulation Scheme with a Differential Coded Data Signal and the Two Carrier Signals

different frequencies (2 FSK) which are turned on and off mutually exclusive according to the binary-coded data signal (see Figure 2.7). In this example, if the data signal is "low" the slower carrier is applied to the transmission antenna and the faster carrier is used if the data signal is "high". Equation 2.2 describes this behavior.

$$u_{FSK}(t) = u_{Carrier0}(t) * u_{signal}(t) + u_{Carrier1}(t) * |u_{signal}(t) - 1| \tag{2.2}$$

The last component that can be used to modulate a carrier signal uses the phase of the signal and is therefore called *Phase-Shift Keying* (abbreviated PSK). For simplicity a phase change of 180 degrees is used in Figure 2.8 which is called 2 PSK or Binary PSK because two phase states can be differentiated. In practice the carrier signal's phase can be shifted by other angles which can be used to encode a sequence of bits which increases the data rate. QPSK (quadruple) uses four phase states for instance, to modulate two bits at the same time, OPSK (octal) modulates three bits, etcetera. Figure 2.9 shows an example of different PSK types with evenly distributed phase states. The symbol associations are

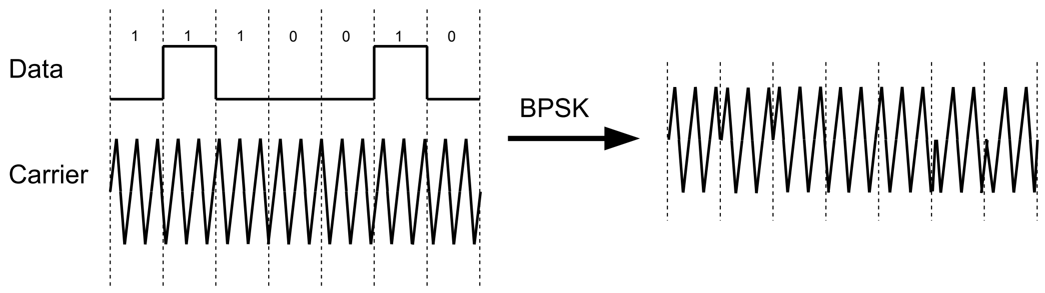


Figure 2.8: PSK Modulation Scheme with a Differential Coded Data Signal and a Carrier with 180 Degrees Phase Shift (BPSK)

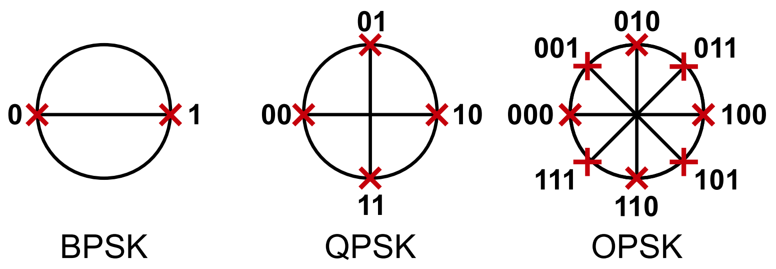


Figure 2.9: Different PSK Types with Even Distribution of Phase States and Exemplified Symbol Association

also just examples and can vary in practice.

Considering the frequency spectrum of ASK, FSK or PSK it lacks the fact that the data-signal spectral lines are placed directly next to the carrier frequency. This can be changed by using another modulation type which differs from the other mentioned types and is called *Modulation with Subcarrier*. Usually it is used for inductively coupled RFID systems like the typical 13.56 MHz HF tags and works with a multi-step modulation system. The encoded data signal is modulated (ASK, FSK or PSK) on a subcarrier that is usually derived from the operating frequency of the carrier by dividing the carrier with a divisor that is a power of two (e.g., 16, 32, 64). On the next stage the output signal of the previous modulation step is modulated on the actual carrier by switching a load resistor according to the modulated subcarrier signal. The result is a higher gap between the spectral lines of the carrier and the data signal. For the demodulation process the antenna signal can be low-pass filtered to separate the carrier from the sub-carrier signal. The whole procedure is shown in Figure 2.10 where an ASK modulation of the data signal is used for the first stage.

Probabilistic versus Deterministic Inventory: The inventory process is a protocol

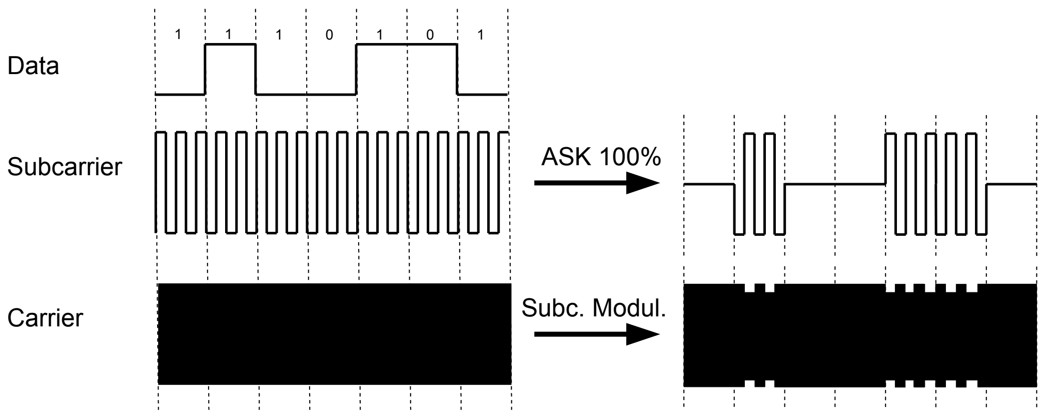


Figure 2.10: Subcarrier Modulation Scheme with a Differential Coded Data Signal and 100% ASK Modulation on the First Stage

feature that is used to search for tags inside a reader field. Since, only one tag can answer at a time there needs to be a mechanism implemented that manages the time slots. For inventory purposes it is also necessary for the reader to detect collisions (cf. Modulation Types). Basically, there exist two principles how the inventory process can be implemented. *Deterministic Inventories* progressively search the UID (a unique number that identifies the tag) space until the whole UID is found and selected for further communication. Therefore the reader sends a start command and all tags in the field return a part or the whole UID at the same time. If the reader detects a collision, it recognizes that from this point on the received data is not consistent anymore and decides to use either a “1” or a “0” for this bit. Then the reader selects all tags that begin with this bit string to be allowed to continue with the inventory process. The tags then send the next part of the UID until another collision is detected and solved. This procedure continues until a whole ID is extracted. One protocol that uses this inventory type is ISO/IEC 14443-3. Figure 2.11 gives an example of the procedure. The reader begins to send the “Start Inventory” command which is received by two tags in the field. Then the tags reply with their UID by synchronously sending the bit sequence of the UID. The first three bits “110” are transmitted successfully because these three bits match within both IDs. On the next bit a collision occurs because the tag on the left sends a “1” and the tag on the right a “0”. In order to solve the collision the reader randomly chooses a “0” for the bit that caused the collision and selects all tags which have IDs that begin with “1100”. This command excludes the tag on the left from the current inventory process and allows the other tag to send its full ID. Depending on the number of tags in the field a collision could theoretically happen with every bit that is transmitted by the tags during the inventory process, but the number of collisions is limited to the number of bits that are used for the ID. That is the reason why this inventory is called deterministic.

The other inventory type is called *Probabilistic Inventory* because collisions are not solved in a deterministic way. Instead, there is a mechanism implemented that decreases the probability of another collision. Since the number of tags in the reader field cannot be

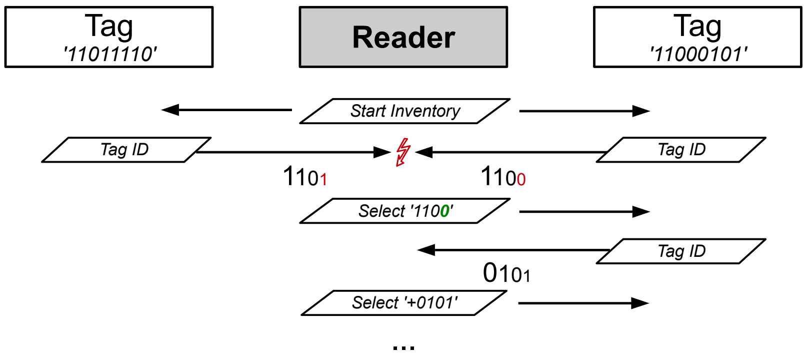


Figure 2.11: Example for a Deterministic Inventory Routine

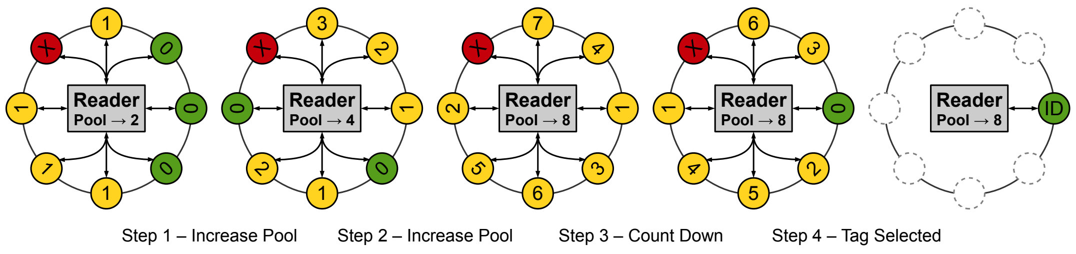


Figure 2.12: Example for a Probabilistic Inventory Routine

detected per se, the reader needs to successively adjust the probability of tag collisions. Also the tags need to have some kind of randomization functionality implemented (cf. Random Number Generator) to pick a random number from a reader-defined pool that selects when the tag will reply. In the following an example for a probabilistic inventory which works similar to the EPC Gen 2 inventory is discussed. At the beginning of the inventory the reader chooses the pool size (e.g., 2 in Figure 2.12) and the tags calculate a random number within the pool range (''0'' or "1"). One tag marked with an "X" was already dismissed from the inventory because of a pre-selection process. All tags that calculated a zero answer with the UID. The reader detects a collision and doubles the pool size for the next round. Therefore, the tags calculate new random numbers. In the following round another collision is detected and the reader doubles the pool size again. Since no tag calculated a zero, no tag answers this time. Instead of shrinking the pool variable the reader executes the inventory with the same settings. This is recognized by the tags which do not recalculate a new random number, but decrease the actual number by one. Finally, only one tag calculated a zero and answers with an ID which is then selected by the reader for further communication. The maximum time for a successful inventory for a given number of tags can only be statistically estimated. There is the theoretical probability that this inventory process will last forever but increasing the pool variable makes this unlikely.

Security and Privacy Features: Security and privacy features are usually not a part of the RFID protocol pers se, but protocols provide ways to implement functionality that is not defined in the standard. The ISO/IEC 14443 protocol provides so-called custom commands which allow companies to implement their own commands. In the EPC Gen 2 protocol there are already some command skeletons defined that can be used to implement cryptographic functionality based on cryptographic suits that can either be standardized or company specific. The type of algorithm that is used for cryptography depends on the application, the price of the tag, the power consumption constraints, the maximum execution time, etcetera. Standardized security algorithms are typically used for smartcards like credit cards where a high level of security is needed and the price for the smartcard is not that relevant as it is, e.g., for a logistic label. Asymmetric algorithms such as elliptic curve cryptography or RSA are high computational problems which also consume more chip size than a light-weight cipher like Present (see Bogdanov et al. [8]). Since, it is not realistic to predict that asymmetric algorithms will be used for cheap products like RFID logistic labels in the next years, there is also the need for low-cost ciphers. Such ciphers provide relatively high security and consume less chip size, execution time and power than asymmetric algorithms. Besides the used cryptographic algorithm also the hardware implementation and the protocols that use the algorithms need to be secure. Many systems are not broken because of the used algorithms but because of the way they were implemented which often allow side-channel attacks, fault attacks or other implementation attacks. With additional hardware effort side channels can be closed and fault attacks can be detected and prevented. Since the complexity of algorithms cannot be infinite every cryptographic implementation can be somehow broken with enough effort and time—whether directly or indirectly. So it is always a trade-off between a high level of security and the implementation effort. This trade-off has to be considered together with the environment where the product is used in order to find the sweet-spot that makes the effort for breaking a system uninteresting for attackers, and the costs low enough to be acceptable. The use of standardized authentication protocols ensures that no mistakes on the protocol layer are made which could offer a vulnerability to man-in-the-middle attacks like replay or relay attacks. Privacy aspects are not fully covered with the use of an encrypted communication or authentication protocol. Also the identity of the tag should stay unknown for unauthorized persons to prevent unwanted tracking which is one of the major privacy concerns. The problem gets more obvious by looking at inventory mechanisms of protocols like ISO/IEC 14443 where the tag's UID is transmitted in plain text form and without previously creating a secured communication channel. Solutions for this problem are discussed in Chapter 3 in Section 3.4.

2.1.3 System-Layer Differentiation

Types of RFID Transponders: RFID transponders can be classified as *1-bit Transponders*, *Labels*, *Tags* or *Smartcards*. A 1-bit transponder is typically used for electronic article surveillance (EAS) where no real information is exchanged between reader and transponder besides the information whether the transponder is in the field or not, or deactivated. The costs for such transponders are very low, because of the high quantities and the simple circuit design.

The next category is called *Labels*, because they consist of a thin and flexible carrier material that is almost invulnerable against physical deformation. They are often used for logistic purposes and therefore produced in very high quantities which makes them very

Property	**1-bit Transp.**	**Labels**	**Tags**	**Smartcards**
Application Example	EAS	logistics	transportation	credit cards
Complexity & Costs	very low	low	medium	high
Logic Implementation	analog only	FSM	FSM	FSM & μC
Security Effort	none	low	medium	high
Quantities	very high	very high	high	medium - low
Power Consumption	very low	low	medium	high

Table 2.2: Typical Properties of RFID Transponder Types

cheap. Since the complexity of the implemented functionality (RFID protocol) is usually quite low, the control logic is implemented as a finite-state machine.
Tags usually provide more complex functionality than labels which raises the costs. Increased functionality normally results in a higher power consumption. One classical tag application are tickets in transportation systems. The security effort is also higher compared to labels, because the economical and reputational damage caused by a cracked system that is used for ticketing is significant.
The most complex RFID transponder systems are *Smartcards*, e.g., credit cards which often use microcontrollers or even have some kind of operating system that manages the system components. Those systems often use standardized cryptography for authentication and secure communication purposes. The price for these transponders is relatively high, because the quantities are much lower than for labels, also the development and manufacturing costs are higher but smartcards usually have a long life cycle. Table 2.2 gives a summarized overview of the typical properties of different RFID types.

Control Logic Differentiation: The way the control logic of the RFID tags is realized is another differentiation feature which is related to the transponder type. The simplest form—without any logic—is used for 1-bit transponders (see Figure 2.13, left) that uses an LC resonant circuit for example to influence the reader field. If the transponder is inside the oscillating reader field the current that is induced in the transponder coil creates a field that works in the opposite direction of the reader field. The resulting magnetic field weakens the reader field which can be recognized by the reader and can be used to decide if the tag is inside the field or not. For transponders that use a communication protocol like EPC Gen 2, a digital control logic is usually implemented in a finite-state machine (abbreviated FSM) style. These transponders can give dynamic answers and transmit data that depends on the commands that were received from the reader (cf. Mario Cardullo's "TRANSPONDER APPARATUS AND SYSTEM" from 1973, Figure 1.3). More dedicated transponders such as smartcards also use microcontroller-based implementations to handle the high complexity that comes with modern protocols, the cryptographic requirements, additional interfaces, etcetera. The difference to the FSM-based designs is that the development process is not only focused on hardware but also on software design (hardware/software co-design) which allows to implement functionality more quickly and on a higher abstraction level. Compared to pure hardware designs the microcontroller-based designs tend to be bigger (it can be shown that when the tag's complexity is high enough a trend reversal takes place, see [22] page 13), less energy efficient and slower.

Formats and Housings: The applications for RFID technology are numerous and so

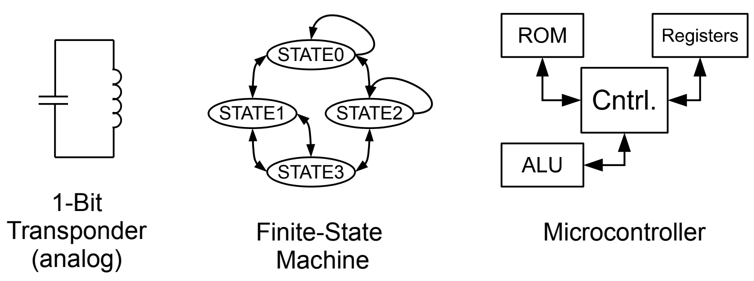


Figure 2.13: Different Realizations of Tag Logic

are the formats (or housings) of the transponders. Figure 2.14 shows some examples of different transponder formats which are already in use. The blue coin in the top left corner is used in public transportation as access token to subway stations in Taipei City, Taiwan. In the upper middle of the figure there is an HF label as it is used for example in libraries to automatically identify books and connect them with information that is stored in a database. Another example (bottom left) shows a UHF tag which is used in logistic systems. The "e-card" was introduced in Austria in 2005 as an electronic form of an insurance card, but the card can also be used to electronically sign or encrypt documents, for e-banking or for online filing of tax returns. Even passports use RFID technology to store and transmit personal details like name, age, home country, biometric data, a picture of the owner, etcetera.

2.2 Electromagnetic Waves as a Communication Medium

At the beginning of Chapter 1 the discovery of the electromagnetic waves was shown from a historical perspective. Furthermore, the relation between the magnetic field and the electric field was mentioned and that the speed of the electromagnetic waves is equal to the speed of light which is defined to be exactly 299,792,458 m/s in the vacuum (see NIST/CODATA [45]). In the following it is explained how electromagnetic waves as they are used for UHF tags can be generated and used for communication purposes.

2.2.1 Generation of Electromagnetic Waves

The limited propagation speed of changes in the electromagnetic field from the source of the field to another point in the space leads to the characteristic wavelike propagation. When an evenly oscillating electromagnetic field is examined, the distance (wavelength λ) between two peaks in the wave is in direct relation to the propagation speed of the wave and inversely proportional to the oscillator frequency (see Equation 2.3 and Figure 2.15). The angle between the $\vec{E}$ field and the $\vec{H}$ field is a right angle. The direction of the field vectors can be figured out according to the so-called "left-hand rule", where the middle finger is the vector of the propagation direction, the index finger is the $\vec{E}$ field and the thumb is the $\vec{H}$ field vector.

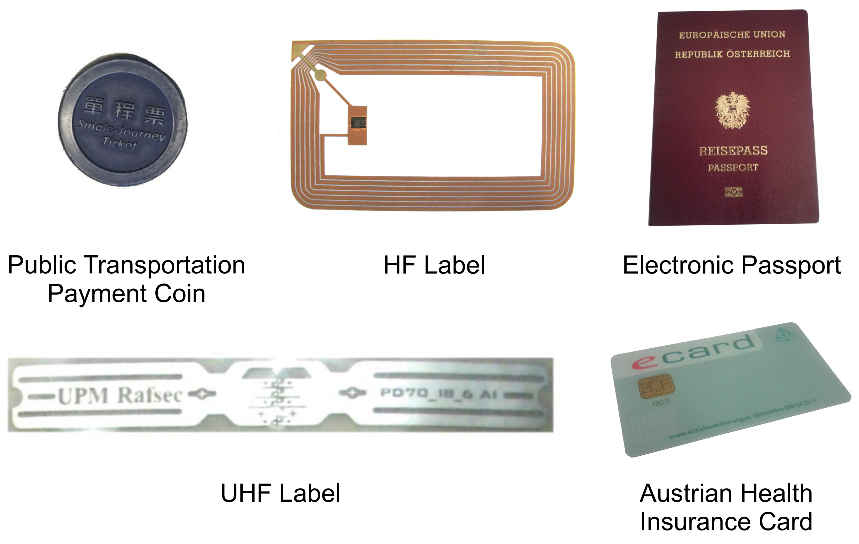

Public Transportation Payment Coin
HF Label
Electronic Passport
UHF Label
Austrian Health Insurance Card

Figure 2.14: Examples for RFID Transponder Formats

$$\lambda = \frac{c}{f} = \frac{[\frac{m}{s}]}{[s^{-1}]} = [m] \tag{2.3}$$

To understand the creation of an electromagnetic wave it might be helpful to look at a dipole antenna that gets charged by an alternating voltage source (see Figure 2.16 which was adapted from Finkenzeller [16] on page 111, Figure 4.58). The dipole antenna can be seen as an unfold capacitor [2] that is connected to an alternating voltage source (cosine source for further explanations) that creates an $\vec{E}$ field between the two plates of the capacitor (Figure 2.16a - c). Figure 2.16d shows the dipole antenna with positive charge carriers at the top of the antenna and negative carriers at the bottom. When assuming that the electric field is created at the time $t = 0$ where the AC cosine source reaches its positive maximum, then it can be said that this is the beginning of the creation process of the electromagnetic field. By changing the voltage direction of the AC source the charge carriers start to move from one side of the antenna to the other—in other words a current flows. This also changes the generated electric field like it is shown in Figure 2.16e. Meanwhile, the propagation of the already generated field lines continue. At a fourth of the AC-source oscillation period the voltage has reached the zero crossover point (Figure 2.16f). This means that no charge separation exists at that point and therefore also no electric field. Instead the field line ends begin to connect and form a closed whirl. Depending on the frequency of the antenna voltage source the whirls start to disconnect from the emitter at a certain point (between Figure 2.16f and g), which is the beginning of the so-called *far field* where the electromagnetic radiation begins. The higher the frequency of the electromagnetic wave is, the smaller the wavelength and the smaller the distance to the emitter of the electromagnetic radiation which implies higher field strengths. Figure 2.16g shows an already disconnected whirl and the beginning of a new whirl that has an opposite field orientation. The distance between two whirls is half the wavelength and so

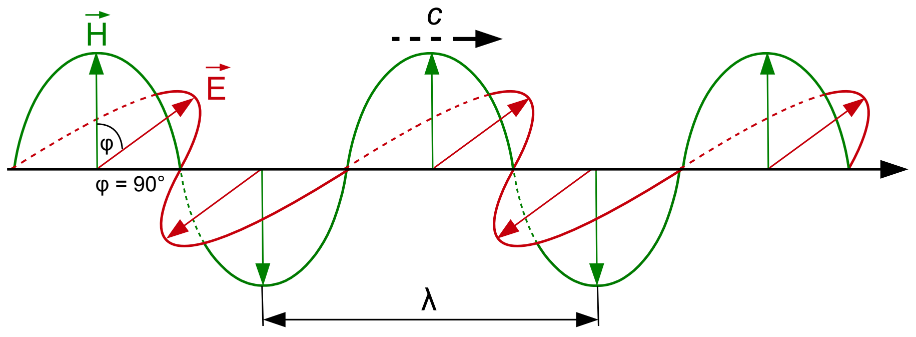


Figure 2.15: Propagation of an Electromagnetic Wave with Constant Frequency and Wavelength

the distance between two whirls with the same field orientation is the wavelength. At the half period of the voltage cosine source the Figure 2.16h is quite similar to the starting point in Figure 2.16d except the orientation of the field. The new generated whirl pushes the already existing whirl away from the transmitter at the speed of light.

2.2.2 Reflection of Electromagnetic Waves

The area between the antenna—where the electromagnetic field is formed—to the point where the radiation of the electromagnetic waves begins (around $\lambda/2\pi$) is called *near field*. Beyond this point the *far field* begins where the electromagnetic waves lost the connection to the antenna and propagate freely in space. Since the transition from near field to far field depends on the wavelength λ and the strength of the field decreases quadratically with the distance from the antenna, only high-frequency systems (UHF and microwave) use electromagnetic waves as communication medium for RFID purposes.
The fact that objects reflect electromagnetic waves is already known since the invention of radar technology (cf. Robert Watson-Watt in Chapter 1). How well an object reflects electromagnetic waves is described by the so-called radar cross-section factor σ which depends on the material (metal reflects better than wood for example), size, shape, surface structure, wavelength, polarization of the electromagnetic wave, etcetera. The influence of the wavelength and the object dimension are very important, therefore the size of the object (in the following called “s”) in relation to the used wavelength is classified into three categories (see Finkenzeller [16], page 116):

Raleigh range ($\lambda >> s$): Has no practical relevance for RFID tags since the radar cross-section factor is too low.

Resonance range ($\lambda \sim s$): An object that is in resonance with the emitted wavelength gets a resonance step-up of σ which is used and wanted for antennas. When the object size or the wavelength is changed just a little around the resonance point resonance step-up is lost.

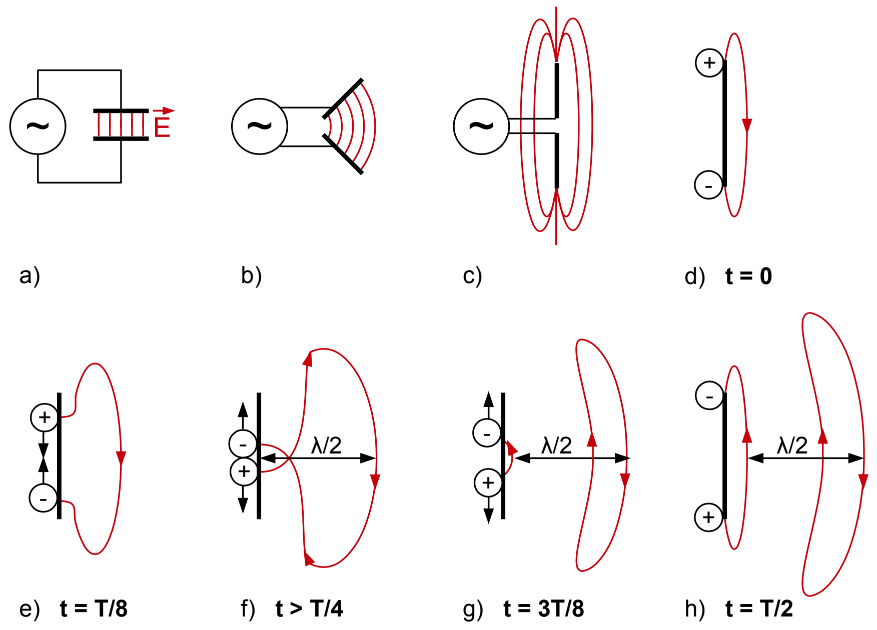


Figure 2.16: Generation of Electromagnetic Waves with a Dipole Antenna

Optical range ($\lambda << s$): The influence of the wavelength to σ is not significant when the dimension of the object is high compared to the wavelength. How well the object reflects the electromagnetic waves then only depends on the shape and orientation of the object.

The propagation of the electromagnetic wave usually is not homogeneous for all directions—as it would be for a so-called isotropic emitter where the radiation pattern is spherical—and depends on the form of the antenna. A dipole for example emits the maximum power at a right angle away from the middle of the antenna (see Figure 2.17) and does not radiate at all upwards or downwards. Other antenna designs increase this directional effect which allows communication over longer distances. To express the relation of the transmission power and the directional effect for different antennas they are compared to an isotropic emitter. The difference of the power transmitted by an isotropic emitter and the power transmitted by another antenna with the same power supply (effective isotropic radiated power, abbreviated EIRP) in the main radiation direction (direction with the maximum power transmission) is expressed by the so-called *antenna-gain* factor "G_i" (see Equation 2.4). For a dipole antenna the G_i factor is around 1.639 according to Finkenzeller, which means that the power supply of the antenna needs to be just about 61% to achieve the same effect as for an isotropic emitter.

$$P_{EIRP} = P_{antenna\_supply} * G_i \tag{2.4}$$

When considering the other side of the communication channel it is important to know

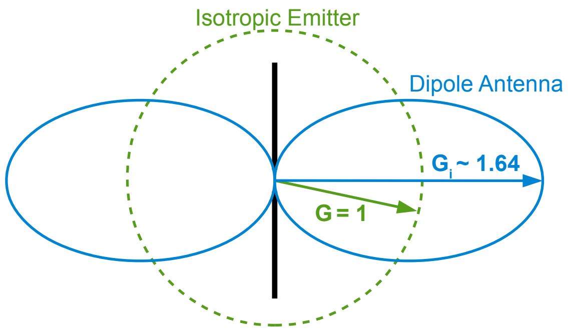


Figure 2.17: Directional Power Emission of a Dipole Antenna Compared to an Isotropic Emitter

which portion of the power that is emitted by the reader can be used by the receiver. The power shrinks with the distance from the emitter according to Equation 2.5 where "S" is the so-called power density (see Figure 2.18). In order to calculate the power that is received by an antenna (in optimal orientation to the reader field) for a given density a proportional factor needs to be introduced to express how much of the incoming field is captured. By taking a look at the dimension of the density (W/m) one will notice that this factor needs to be an area which is called the effective aperture "A_e". For illustration purposes A_e can be seen as an area where a certain amount of the emitted electromagnetic flux—that is described by its density—gets through and generates a certain amount of power (see Equation 2.6). This power is then converted into an antenna voltage which is partially absorbed by the circuit connected to the antenna and the rest is reflected. The amount of reflected energy can also be expressed as an area called scatter aperture A_s which equals the radar cross-section factor σ. It can be shown that for a power matched antenna A_e equals A_s. In this case the amount of reflected energy is the same as the amount of energy that can effectively be used by the receiver (see Finkenzeller [16], page 120). To achieve an optimal matched antenna the circuit behind the antenna (analog front-end) has to compensate the antenna's complex impedance. Therefore the analog front-end's input impedance needs to be the complex conjugated impedance of the antenna's equivalent circuit.

$$S = \frac{P_{EIRP}}{4\pi * r^2} \tag{2.5}$$

$$P_e = A_e * S \tag{2.6}$$

For communication purposes the tag varies the real part (resonator) or the complex part (capacitor) of the matching network by switching an additional resonator or capacitor which changes the amount of reflected energy. The corner cases for the reflected energy can be found by a short-circuit of the antenna terminals—for a considered loss-less antenna circuit ($X_{ant.} = X_c$)—where the maximum power is reflected with $\sigma = 4 * A_e$ and the open loop case where no energy is reflected et all. The backscattered power that is used for the modulation from tag to reader can lie between those boundaries. For an example of the analog front-end with a backscatter modulator see Section 5.1.7.

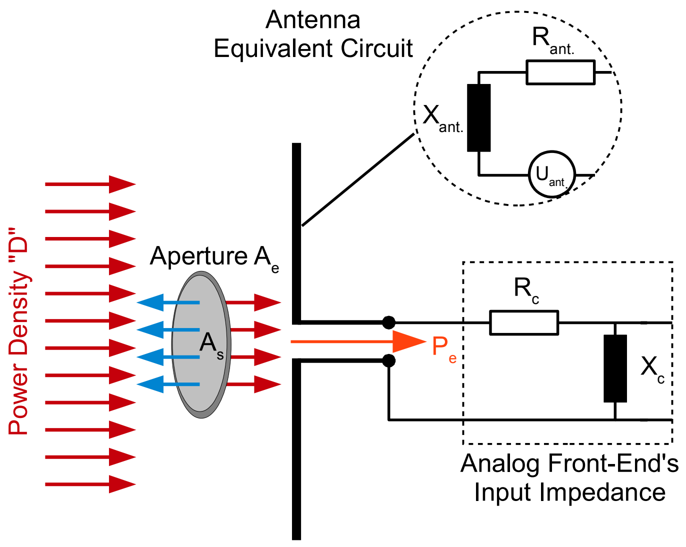


Figure 2.18: Receiver Antenna with Power Density of the Reader Field, Antenna Apertures A_e and A_s, and Equivalent Circuits of the Antenna and the Analog Front-End of the Tag

2.3 EPC Gen 2 as an Example of a Modern RFID Protocol

In 2003 the EPCglobal Inc. was founded which developed the second version of the EPC standard (see Section 1). The EPC Gen 2 standard is a UHF (860 - 960 MHz) protocol focusing on RFID applications that works with electronic product codes (abbreviated EPC), which was more or less treated as a pure electronic replacement of the bar-code system at the beginning of the EPC. Due to optional and extendible protocol functionality this standard is also suitable for more complex applications. In the following an overview of the functionality of the protocol with focus on the basic principles is given. Section 2.3.1 begins with an explanation of the EPC number, the tag's finite-state machine and the different memory banks of the tag. In the next sections a step-by-step walk through the reader access of a tag is explained, beginning with the selection of a tag population (Section 2.3.2) for the subsequent inventory process (Section 2.3.3) that is used to prepare the access of a single tag. Finally, different ways to access tag functionality are discussed. The lowest abstraction layer (physical layer) is not discussed since the sub-carrier modulation, different modulation types, data encoding, the backscattering principle, etcetera are already discussed in Section 2. For a more detailed description see the EPC Gen 2 standard provided on the GS1 homepage [24].

2.3.1 EPC Gen 2 Basics

The electronic product code (abbreviated EPC) is a unique number which identifies, e.g., a product that is attached to the tag (label). In contrast to barcodes like EAN/UCC the

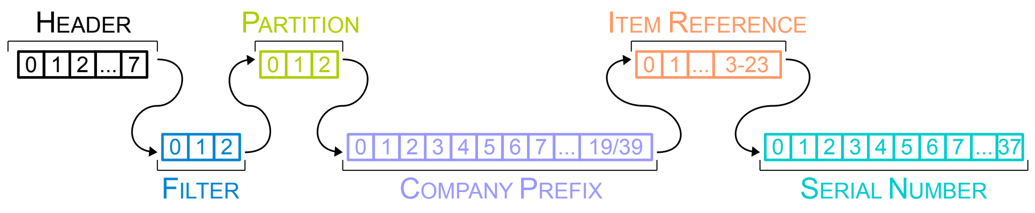


Figure 2.19: EPC Encoding According to SGTIN-96

EPC can also be used to identify an individual product instead of the article by using a consecutive number. There is not a single encoding type for EPC but about 20 different encoding types which are defined in the GS1 Tag Data Standard [23]. The structure of the encodings looks quite similar (see Figure 2.19 for an example) which always begins with a *Header* field that defines the used encoding type. In the given SGTIN-96 example (abbreviation for Serialized Global Trade Item Number with 96 bits length) the header is followed by a three-bit *Filter* field which allows a preselection of the tags (cf. "Select" in Section 2.3.2) on different packaging levels. Since the length of *Company Prefix* is between 20 to 40 bits, the *Item Reference* field length is between 4 and 24 bits and the overall length must be 44 bits, the *Partition* field defines the length of the *Company Prefix* respectively the border to *Item Reference*. The *Company Prefix* identifies the so-called "EPCglobal Manager" which is usually the company that manufactured the product. Similar to barcodes the *Item Reference* number defines the article but in addition a unique *Serial Number* is used to identify an individual product. According to the GS1 Tag Data Standard 1.6 [23] there exist EPC numbers up to 202 bits besides the formal used 64 and 96-bit EPC numbers.

Finite State Machine of an EPC Gen 2 Tag

In order to understand the explanations in the next sections a basic knowledge about the finite-state machine (abbreviated FSM) that is the basement of the EPC Gen 2 "tag-identification layer" might be helpful. Figure 2.20 shows the FSM that is subdivided into three phases "Select", "Inventory" and "Access". All phases have states belonging to them and every state has specific commands that are mainly connected with them. However, some commands are valid in more than one state like a "Query" which starts a new inventory round independently from the actual tag state.
The **Select** phase begins with the *POWER UP* state—which actually is not a defined EPC Gen 2 state—that is automatically entered when the tag is powered by the reader field. If the tag was not killed before (the tag accepts no commands and does not respond in this state) the *READY* state is entered. In this state a preselection of the tag population is performed which is then part of the **Inventory** phase. This phase is started by a "Query" command of the reader that contains the settings for the probabilistic inventory. Depending on the result of the command the tag enters either the *ARBITRATE* state where the tag waits for another inventory command or the *REPLY* state. In the latter case, the tag backscatters a number and waits for the readers acknowledge ("ACK"). When the reader requests another random number by sending a "Req_RN" the tag answers with a number

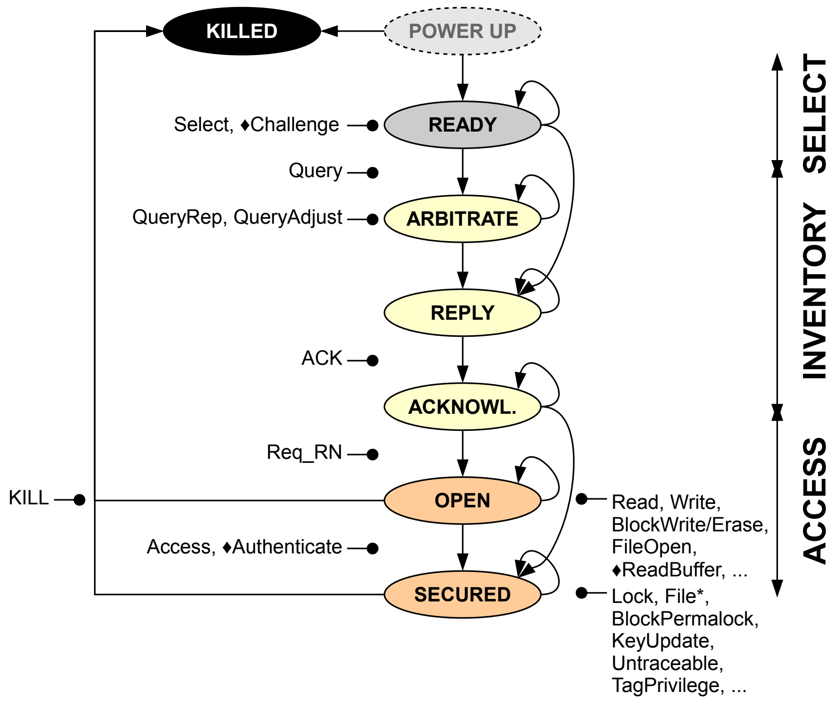


Figure 2.20: EPC Gen 2 Finite-State Machine Subdivided into “Select”, “Inventory” and “Access” Phases and the According Commands. Security-Enhanced Commands are Marked with a Diamond Symbol

that becomes the “handle” of the tag (a unique identifier used in future reader commands to address single tags) and enters the **Access** phase. If the tag is security-enhanced the next state is *OPEN*, else the *SECURED* state is entered directly. The first state allows only restricted access to the stored tag information, memory and commands until some kind of security procedure is successfully executed. Then the tag enters the *SECURED* state with more privileges that depend on the actual implementation of the tag. Killing a tag with the “KILL” command is possible from both states if supported.

Tag Memory Banks

The tag memory contains up to four separate memory banks as shown in Figure 2.21. The **Reserved** memory bank contains the optional “KILL” and “ACCESS” passwords which can be used to bring the tag into the permanent *KILLED* state or from the *OPEN* to the *SECURED* state (see Figure 2.20). If there is no kill password set or if it contains only zeros then the tag cannot be killed. An unimplemented access password leads to a direct transition from *ACKNOWLEDGED* to the *SECURED* state.
The **EPC** memory bank stores besides the EPC number also a so-called “StoredCRC” that is a cyclic redundancy checksum (CRC-16) over the EPC memory bank without the optional “XPC”. Furthermore, the protocol-control words “StoredPC” and its optional extension the “XPC” are located in this memory bank. The protocol-control words store information such as the length of the EPC number, flags that indicate that a user memory

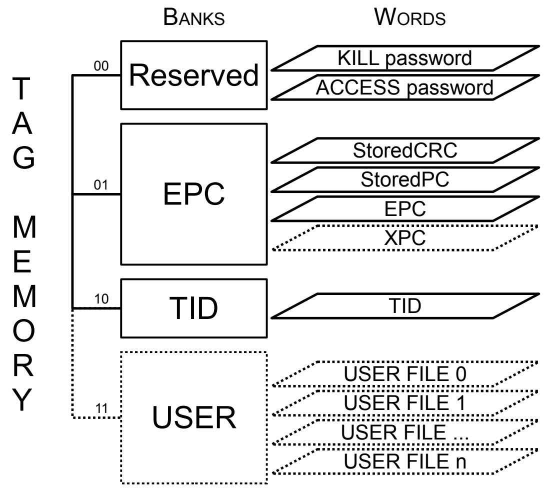


Figure 2.21: EPC Gen 2 Memory Separated into Banks and Words

bank or the XPC is available, a flag that signals that an answer to a Challenge command is available and stored in the ResponseBuffer (see Section 2.3.2), etcetera.
In the **TID** there is an allocation class identifier stored according to ISO/IEC 15963 which is an 8-bit number that defines the type of the tag followed by information that is used to identify the tag's functionality like custom commands or optional features.
The **USER** memory is optional and contains application-specific files that can be accessed by file commands and after the mapping into the USER memory used like other memory banks.

2.3.2 Selection Phase

This phase is automatically entered when tags enter a reader field (tags that are not already killed) and serves as a preselection process for the inventory procedure. Therefore the tags can be selected or deselected by **Select** commands that contain a comparison criterion which the tags either fulfill or not. Depending on the result of the comparison the tag performs an action that is also defined by the Select command and can either be a union ∪, an intersection ∩ or a negation operation. By applying subsequent Selects a subset of the available tags can be chosen that match complex criteria. The tags implement four different inventory sessions (called S0 - S3) which can have the value "A" or "B" and are also-called "inventoried" flags (see Figure 2.22a) . Every inventory round addresses only one session at a time. The presence of more than one session gets useful if more than one reader inventories a tag population. Each reader works then on a different session. Another flag is the "SL" or "selected" flag which is available for all sessions and can be used optionally for the inventory in connection with a session flag.
Figure 2.22 (b - d) shows the use of session flags for the selection phase in a single-reader environment. At the beginning (b) all tags in the reader field are powered and initialize their session flags by setting them to "A"—only one session flag is shown in the figure for each tag. The blue circle shows the first selection criterion that is applied in the

next figure to set all tags inside the circle (c) to "A" and the others to "B". For a more complex selection and to demonstrate the different actions that can be performed on a tag population subset another selection criterion (orange circle) is used for 2.22d2 - 3. In Figure 2.22d1 the second criterion is not used, but a "Select" command to invert the selection from the previous figure is used instead. Inversion which is the same as a logic NOT ($\neg Criterion1$) is very useful since some criteria are more easily defined by their opposite. The next figure shows the union ($\cup$) function that connects both criteria by a logic OR ($Criterion1 \vee Criterion2$) which is like a graphical melting of both criterion circles. In the last figure the intersection function ($\cap$) is used which works as a logic AND ($Criterion1 \wedge Criterion2$) and selects all tags that are shared by both criterion circles. By using the logic functions AND, OR and NOT consecutively all logic operations can be achieved.
The selection criteria can address different memory banks (see Figure 2.21) except the "Reserved" bank. For example by addressing the "EPC" only tags from a certain manufacture can be selected or tags with special abilities which are stored in the PC and XPC control words. The XPC word also contains an indicator called "C" flag that is important for the second "Selection"-phase command **Challenge** which is optional and tells a reader that a challenge answer was calculated and stored. The Challenge command is a skeleton frame which only defines some basic fields, but the real functionality is defined by the selected Cryptographic Suite (abbreviated CS) which can be either standardized or defined by the manufacturer of the tag. By sending a Challenge command the tags that are capable of the defined CS calculate an answer and either store it in the "ResponseBuffer" or backscatter it after the readers ACK in the "REPLY" state. If the answer is stored the "C" flag is asserted which can then be selected by the reader to only inventory the tags that calculated an answer. After the inventory the reader can perform a read-out of the ResponseBuffer by sending a "ReadBuffer" command and check the correctness of the answer (see Chapter 3 for details).

2.3.3 Inventory Phase

In this phase the reader wants so detect which tags are in the field—that fulfill the constraints that were set in the "Selection" phase—and prepare them for the "ACCESS" phase were only one tag at a time is addressed. The goal is to detect a single tag and connect a "handle" with it that is used lateron as a reference to send commands to a specific tag. The principle of the probabilistic inventory is already explained in Section 2.1.2. Therefore only the EPC-specific functions are explained in the following.
The "Query" command starts a new inventory round. Besides some physical-layer parameters (data rate, usage of a pilot tone, coding), the Query defines also the Session (S0 - S3) that is used, the target ("A" or "B"), the usage of the SL flag and the pool size ("Q" parameter) of the probabilistic inventory. Depending on the "Q" value all tags pick a random number inside the pool size which is a number between zero and 2^Q-1 that is loaded into the "slot counter". Tags that calculated a zero transit automatically to the "REPLY" state and backscatter a 16-bit random number. All other tags transit to the "ARBITRATE" state. If collisions occur then the reader can increase the pool size by sending a "QueryAdjust" command and increase the "Q" value. Decreasing is also possible but not useful when too many tags have already answered. If no tags answer a "QueryRep" command is used to decrement the slot counter until one or more tags reach a

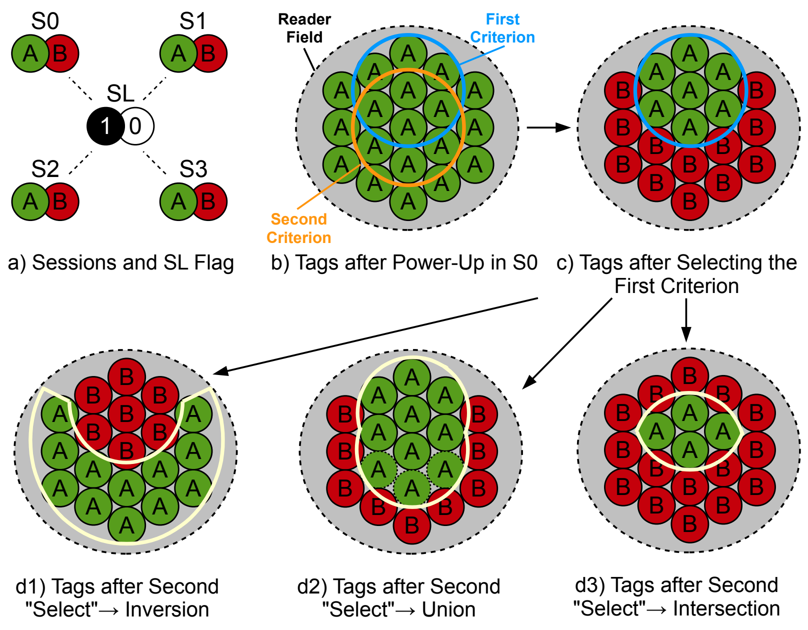

S0
S1
SL
1
0
S2
S3
Reader Field
First Criterion
Second Criterion
a) Sessions and SL Flag
b) Tags after Power-Up in S0
c) Tags after Selecting the First Criterion
d1) Tags after Second "Select"→ Inversion
d2) Tags after Second "Select"→ Union
d3) Tags after Second "Select"→ Intersection

Figure 2.22: Inventory Sessions and SL Flag, and a Consecutive “Select” Procedure

slot counter value of zero and answer the reader. By sending one or more successive Query commands a single tag responds with a 16-bit random number. This is acknowledged by the reader through an “ACK” command that contains the number that the tag sent before. The answer of the tag to the ACK depends on the configuration of the tag and the settings that were transmitted by the reader before and could contain the (X)PC, the EPC, the “Challenge” command response and/or a CRC. Finally, the reader requests another 16-bit random number by sending the “Req_RN” command which then becomes the “handle” of the tag. The tag enters the “OPEN” or “SECURED” state depending on the support of security algorithms. The whole inventory phase without a detailed illustration of the transition from “ARBITRATE” to “REPLY” is shown in Figure 2.23.

2.3.4 Access Phase

The “Access” phase takes place in the “OPEN” and the “SECURED” state and contains five mandatory and 15 optional commands (see Figure 2.24). Furthermore, there can be custom commands implemented that are tag specific or proprietary commands which must not be used for field-deployed tags. Therefore there has to be the possibility to disable these commands (e.g., test commands). The mandatory and optional commands are not all available in the “OPEN” state. Some functions can only be executed in the “SECURED” state, for example memory locking, privilege modifications, key changes, etcetera. To transit to the “SECURED” state, the tag either implements the “Access” command with a valid password in the “Reserved” memory bank, or the “Authenticate” command according to a cryptographic suite. If no security function is implemented the tag directly transfers from the “ACKNOWLEDGED” state to the “SECURED” state.
For the reading and writing of the tag’s memory (see Figure 2.21) there exist the mandatory “Read” and “Write” commands and the optional “Block*” commands that can be

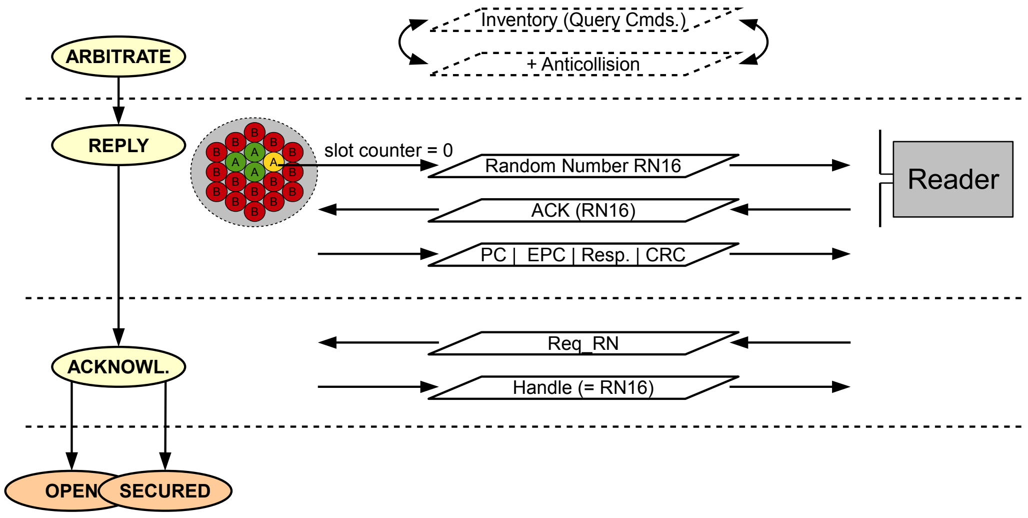


Figure 2.23: Inventory Procedure without Probabilistic Anticollision

used to manipulate a group of subsequent memory words. The memory regions can also have manipulatable privileges (read or write locks) that can be set or made permanent with the “Lock” or “BlockPermalock” command.
All cryptographic commands are optional and their functionality is usually defined in a cryptographic suite except the “Access” command (see Section 2.3.1) and “Untraceable” which allows hiding of the EPC parts from unauthenticated interrogators. The “AuthComm” and “SecureComm” commands need a precede “Authenticate” command and are used to execute a multi-step authentication protocol (see Chapter 3) or for a secured data transfer. It is also allowed to implement multiple security levels with different passwords and privileges that are connected to each other. For example, if a reader with superuser privileges authenticates he could change keys by the “KeyUpdate” command or change privileges with the “TagPrivilege” command which is not permitted for any other security level. The cryptographic suite defines the used algorithms as well as the authentication protocols and available cipher modes. It also defines how secure communication is implemented, how the payload of the “Authenticate” and other commands must look like, etcetera.
The last “Access”-phase related commands are used for files. File access in EPC Gen 2 works memory mapped. Therefore, a specific file can be opened with “FileOpen” and the file number, and then this file is used like any other tag memory by accessing the “USER” memory bank. Files can also be connected to different privileges and could also require a preceding authentication to be fulfilled. The available files can be explored with the “FileList” command which transmits the size, privileges and other attributes. Dedicated tags can use the “FileSetup” command to dynamically change the size of files or their type.

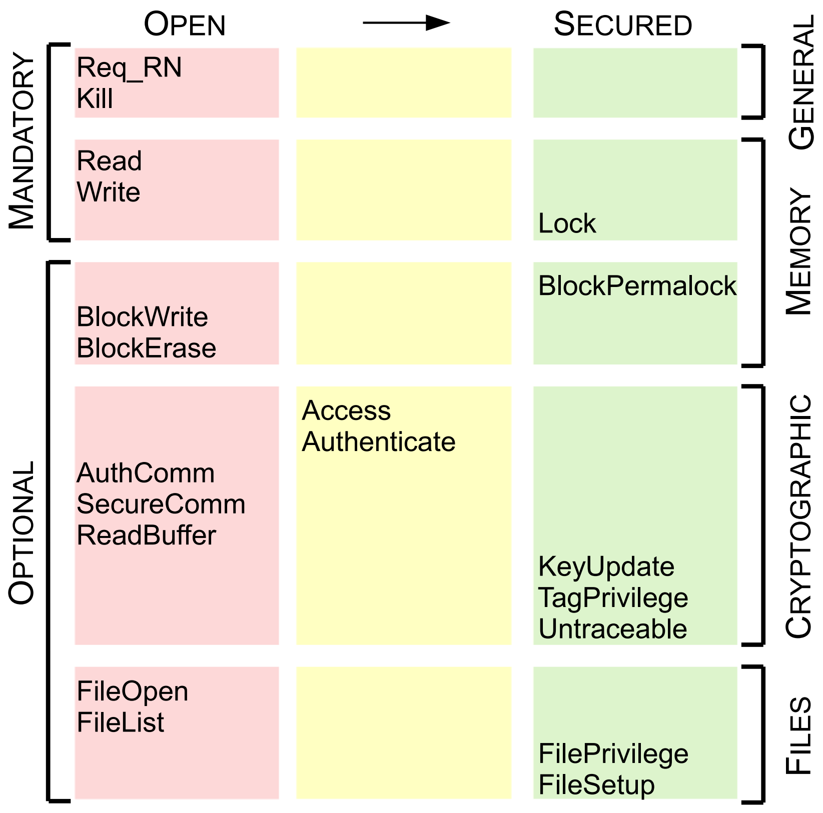


Figure 2.24: EPC Gen 2 Commands Available in the “OPEN” and “SECURED” States

Chapter 3

Authentication Methods and Privacy

In the following, authentication methods for constrained RFID tags based on symmetric-key cryptography are explained and privacy aspects are discussed. The chapter starts with a short introduction on different authenticity-provisioning functions and their suitability for constrained tags in Section 3.1. Afterwards, the Advanced Encryption Standard (abbreviated AES) as an example for a symmetric-key block-cipher is introduced in Section 3.2. In Section 3.3 different types of authentication methods are explained and examples are given. The last section is about privacy and discusses different approaches for RFID systems.

3.1 Authenticity-Provisioning Functions

There are different ways to prove the authenticity of communication parties in digital systems. One is the knowledge of a **shared secret** like a password or a PIN code that is entered to authenticate. The weakness of such a system is that by eavesdropping the communication the system could be broken (Figure 3.1a). There is a better way to use a shared secret for proving authenticity by not directly transmitting the password, but using the password to create a proof of the knowledge of the password. Such a system could rely on **symmetric-key cryptography** like the AES algorithm (Section 3.2) which uses the same password for encrypting and decrypting data. In order to prove the authenticity of one communication party the other one creates a challenge that could only be answered correctly by one that knows the shared secret (Figure 3.1b). Such systems provide a high security level—if they are implemented correctly—and can be implemented efficiently to be used for RFID tags. That is the reason why authentication methods based on symmetric-key algorithms are presented in Section 3.3. A problem that occurs in symmetric-key authentication systems is the distribution of the shared secret. By using **public-key cryptography** a secure communication channel between two parties can be constructed. Public-key cryptography works by having different keys for encrypting and decrypting data. The key that is used for encryption is called public key because it is used by others to encrypt data that could only be read by the owner of the private key (decryption key). The public key does not need to be kept secret. By encrypting the data with the public key it can be ensured that only the holder of the private key can decrypt the message. Public-key cryptography provides so-called digital signatures (Figure 3.1c) to prove the origin of data. However, the key-distribution problem between two parties cannot be solved in a direct way because the secure distribution of a shared secret implies the knowledge of a shared secret (Dent and Mitchell [11], on page 241). Otherwise, the public-key exchange might be corrupted by a third party in the middle that catches the transport messages of the public keys and changes the transported keys. The attacker would then work as a "man in the middle" that could read messages of both parties. Instead the public-key establishment can be done by using a supporting public-key infrastructure (abbreviated PKI) which implies high infrastructural effort and communication effort. Public-key cryptography methods are also very complex computational problems which usually results in a higher chip size and more energy consumption than symmetric-key cryptography. Another way to implement an authentication protocol is to use **zero-knowledge functions**. These functions are similar to public-key cryptography [11] but have an additional property: An attacker that observes the communication cannot learn anything about the secret key. Figure 3.1d shows an example for a zero-knowledge protocol. The person who has to be authenticated ("Bob") enters the room first and goes through the left or right corridor to door "B" behind the wall without being observed by anyone else. Then door "A" is opened and the "observer" looks into the room. He tells Bob to appear on the left or on the right side of door "B" without knowing on which side Bob was before. The chance is 50% that he already stands on the right side and does not need to open door "B" to get to the other side which can only be done by a person who has the right key. When this procedure is repeated and Bob always appears on the correct side then the probability increases ($1 - 0.5^r$) that he is the owner of the correct key to door "B" with every round "r". An eavesdropper could just peek into the room through door "A" and therefore sees just the same as the observer. Protocols that use the zero-knowledge principles are complex and imply a lot of communication effort and therefore they are unsuitable for low-cost tags. ISO/IEC 9798-5 [30] contains different entity authentication mechanisms based on zero-knowledge techniques.

3.2 The Advanced Encryption Standard

In 1997 the National Institute of Standards and Technology (abbreviated NIST) started an international competition on finding a new block-cipher algorithm standard that should become the successor of the outdated DES algorithm (see Dent and Mitchell [11], page 52). The requirements for the so-called Advanced Encryption Standard (abbreviated AES) were a supported block size of 128 bits and the support of 128, 192 and 256 bits of key length. At the beginning of the selection phase there were 22 submitted algorithms, but because not all did fulfill the entry requirements only 15 were considered for the evaluation process. Finally, there were five candidates for the AES. The winner was the "Rijndael" algorithm [10] that was submitted by Joan Daemen and Vincent Rijmen and became the new block-cipher standard in 2001.
The original algorithm supports a variable block size with the values of 128, 192 or 256 bits analog to the key size. For AES only the 128-bit block-size version was used. At the beginning of the encryption, the plain text is stored in the 128-bit "State" (see Figure 3.2) which is a matrix of bytes with four columns and four rows. The key is also stored in a byte matrix with four rows but variable columns (4, 6 or 8). A cipher round can be subdivided into four functions that are sequentially applied on the State.

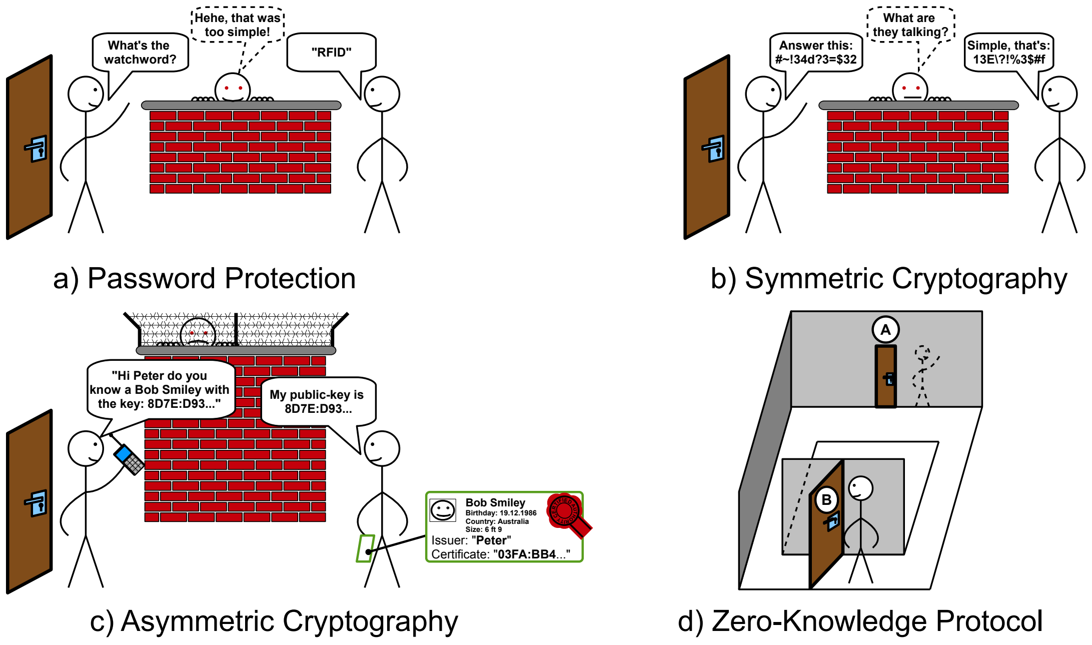

What's the watchword?
Hehe, that was too simple!
"RFID"
a) Password Protection
Answer this: #~!34d?3=$32
What are they talking?
Simple, that's: 13E\?!%3$#f
b) Symmetric Cryptography
"Hi Peter do you know a Bob Smiley with the key: 8D7E:D93..."
My public-key is 8D7E:D93...
Bob Smiley
Birthday: 19.12.1986
Country: Australia
Size: 6 ft 9
Issuer: "Peter"
Certificate: "03FA:BB4..."
CERTIFIED AUTHORITY
c) Asymmetric Cryptography
A
B
d) Zero-Knowledge Protocol

Figure 3.1: Different Ways to Authenticate

The first one is called “SubBytes” and is a nonlinear function that is applied on each byte of the State according to a substitution table. This table is also called S-box and defines a unique output value for every of the 256 possible input values. The substitution is therefore invertible. The next function “ShiftRows” performs a cyclic shift of each State row with different offset which can be seen in Figure 3.3b. In “MixColumns”, a new value $(b_{j,i})$ for each byte of the State matrix $(a_{j,i})$ is calculated by a matrix multiplication according to Equation 3.1. Each State column is interpreted as a $GF(2^8)$ polynomial that is multiplied modulo x^4+1 with another polynomial $3x^3+x^2+x+2$ ($c_{j,i}$, in matrix form).

$$\begin{pmatrix} b_{0,i} \\ b_{1,i} \\ b_{2,i} \\ b_{3,i} \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_{0,i} \\ a_{1,i} \\ a_{2,i} \\ a_{3,i} \end{pmatrix} \tag{3.1}$$

The mathematical form of the MixColumns function hides the elegant simplicity that allows the efficient implementation on computers since only logical shifts and additions are used. Finally, the “AddRoundKey” function is applied on the State. With this function the previously calculated “round key” is connected with the State using a logical XOR ($\oplus$). The number of rounds N_r depends on the chosen key size and is 10 for the 128-bit key, 12 for the 192-bit key and 14 for the 256-bit key. In the last round the “MixColumns” function is left out. Another part of the algorithm is the calculation and selection of the round keys which is also called “Key Schedule”. The calculation of the round keys is done in two steps. The first step is called “Key Expansion” and calculates the key material for the used round keys which is $(N_r + 1) * 128$ bits. In the second step called “Round Key Selection” the key material for the actual round key is chosen. To save memory the key material does not need to be precalculated and can be calculated on-the-fly (one round-key per round is calculated depending on the last one). For decryption the on-the-fly key

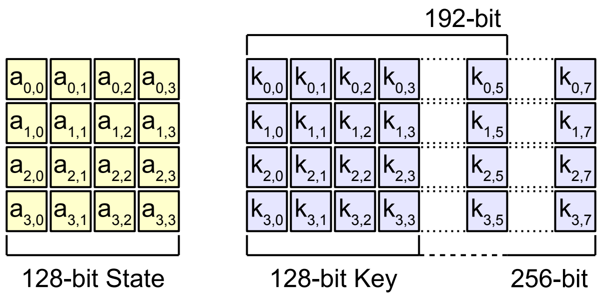


Figure 3.2: AES "State"-Byte Matrix and Variable Key Matrix

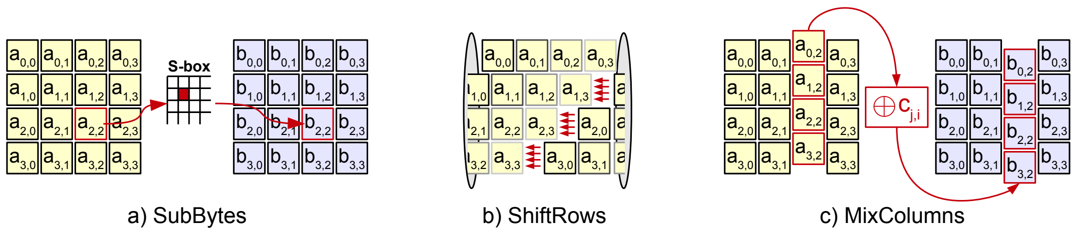


Figure 3.3: AES Round Functions

scheduling is a bit more complex because the keys have to be calculated in the inverse order and the inverse of the MixColumns matrix is also more complex. This makes the AES decryption procedure slower than encryption (cf. Plos et al. [50] on page 122—3,084 cycles for encryption versus 4,505 for decryption on an 8-bit microcontroller).

3.3 Authentication for Symmetric-Key Cryptography

Security is a very complex topic which has to be considered on every design level. The best tested cryptographic algorithm is worthless when the implementation or the protocol that uses this algorithm is insecure. The protocol below shows a very naive and insecure example of a self-made authentication protocol.

$$\begin{aligned} Reader \rightarrow Tag: &\quad AuthRequest \\ Tag \rightarrow Reader: &\quad ENCRYPT(tag\_identity\_string) \end{aligned}$$

First, the reader sends an authentication request to the tag. Upon receiving the request, the tag calculates a response by encrypting a string that uniquely identifies the tag and sends it back to the reader. After extraction and comparison of the "tag_identity_string" with an expected answer, the authentication protocol terminates successfully. The problem gets evident when an attacker eavesdropping the communication is considered. In

that case the attacker could build a tag that always answers to the "AuthRequest" like the eavesdropped tag in order to fake its authenticity (cf. replay attack). In Section 3.1 it was already stated that for constrained RFID tags symmetric-key cryptography with a challenge-response authentication protocol is a good trade-off between security and implementation costs. In the following, some examples of challenge-response authentication protocols are given.
ISO/IEC 9798-2 and 9798-4 contain protocols for unilateral authentication and for mutual authentication. In terms of RFID three different authentication methods can be distinguished. A $Reader \Rightarrow Tag$ authentication is called "Interrogator Authentication", a $Tag \Rightarrow Reader$ authentication is a "Tag Authentication" and $Reader \Leftrightarrow Tag$ is called "Mutual Authentication". Some of the ISO/IEC 9798-* protocols include time stamps to guarantee freshness of the received messages which need synchronized clocks on both sides. Providing these clocks is not trivial even for PCs (see Dent and Mitchell [11], page 179) and unfeasible for RFID tags. Instead so-called "nonces" are used to guarantee freshness of the material. A nonce is usually a randomly chosen number that should only be used once (nonce = **N**umber used **ONCE**). In challenge-response protocols nonces are used for creating the challenge. Therefore, the security of such a protocol depends on the randomness of the created challenge and its uniqueness. The following example of a **Tag-Authentication** protocol was adopted from S. Dominikus et al. [12] which uses derived versions of authentication protocols from the ISO/IEC 9798 standard.

$$\begin{aligned}
Reader \rightarrow Tag: &\quad InventoryRequest \\
Tag \rightarrow Reader: &\quad ID \\
Reader \rightarrow Tag: &\quad AuthRequest \mid\mid ID \mid\mid R_R \\
Tag \rightarrow Reader: &\quad ENCRYPT(R_T \mid\mid R_R) \mid\mid R_T
\end{aligned}$$

After the tag was successfully inventoried and transmitted its ID, the reader sends a request for a tag authentication that includes the tag ID and the nonce R_R (= challenge). The tag encrypts the challenge together with a randomly chosen nonce R_T and sends it back to the reader. The reader checks the correctness of the response by decrypting the message and if the message contains the original challenge then the authentication was successful. One possible attack on this protocol is to eavesdrop the communication and save the challenge-response pairs. When enough different pairs are collected the tag could be cloned. Therefore, the length of the challenge is important to make this kind of attacks uninteresting for attackers. For this authentication method the collection of challenge-response pairs might only be a theoretical attack since the number of trials it takes to find all pairs is at least 2^l (where "l" is the length of the challenge in bits). Furthermore, for cloning the tag an enormous tag memory would be needed. When the other way of unilateral authentication is considered then the attack becomes more interesting for an attacker. In the following, an **Interrogator-Authentication** example is given that was also adopted from S. Dominikus et al. [12].

$$\begin{aligned}
Reader \rightarrow Tag: &\quad InventoryRequest\\
Tag \rightarrow Reader: &\quad R_T\\
Reader \rightarrow Tag: &\quad ReaderAuth \mid\mid R_T \mid\mid ENCRYPT(R_R \mid\mid R_T) \mid\mid R_R\\
Tag \rightarrow Reader: &\quad ID
\end{aligned}$$

In this protocol the tag answers to the inventory with a nonce R_T instead of its ID. The reader uses the nonce for addressing the tag and to calculate a valid response together with another nonce R_R that is created by the reader itself. If the response is decrypted by the tag and contains the correct challenge then the tag replies its ID. The transmission of the plain tag ID is not a good practice in terms of privacy. This problem is faced in the last protocol example in this section that shows a **Mutual Authentication** protocol.

$$\begin{aligned}
Reader \rightarrow Tag: &\quad InventoryRequest\\
Tag \rightarrow Reader: &\quad R_T\\
Reader \rightarrow Tag: &\quad ReaderAuth \mid\mid R_T \mid\mid ENCRYPT(R_R \mid\mid R_T) \mid\mid R_R\\
Tag \rightarrow Reader: &\quad ENCRYPT(R_T \oplus ID \mid\mid R_R)
\end{aligned}$$

This protocol equals the "Interrogator Authentication" example except the last line. Instead of the plain ID the tag uses an XOR to combine R_R with the ID and encrypts the result together with the concatenated nonce R_R. The reader uses the tag's answer to prove its authenticity and to extract the tag ID. In Section 3.4 another authentication protocol is discussed that uses hash functions for authentication and to hide the tag ID.

3.4 Privacy for RFID Tags

Privacy is one of the major concerns of RFID-technology critics and one of the reasons why the impact has not yet become as it was predicted at the beginning of the RFID hype in the early 2000's (Florian Michahelles from ETH Zürich in his presentation on "When will RFID embrace our everyday lifes?" for the RFIDSec 2012 in Nijmegen [18]). The early adopters were the companies Metro and Wal-Mart. Both tried to integrate RFID technology into their logistic systems. Wal-Mart started its RFID program in 2003 with the goal that all pallets from external suppliers should be tagged with RFID labels until 2006. Metro went one step beyond and claimed that all articles in the store would be tagged by 2006. Both programs were shut down because of public campaigns that were concerned that using RFID labels would result in a loss of privacy. This development is not surprising and is well known for technologies that are hyped like RFID technology. The phases of an over-hyped technology are best explained in the Hype-Cycle Diagram in Figure 3.4 that was introduced by Jackie Fenn in 1995 (see Gartner [19] for a detailed explanation). At the beginning the interest in the technology gets pushed by the media and others until the "Peak of Inflated Expectations" is reached. Then there comes a steep slope down to the "Trough of Disillusionment" because it gets evident that the expectations were too high and the technology lacks acceptance. Further developments show

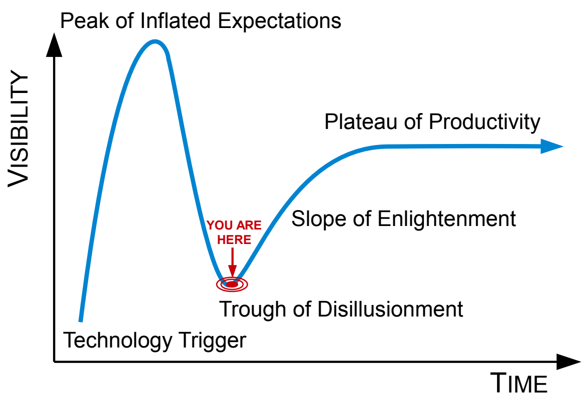


Figure 3.4: Gartner's Hype Cycle

the real benefits of the technology and new fields of applications are discovered ("Slope of Enlightenment"). At the "Plateau of Productivity" the technology has evolved and the products become widely accepted. Furthermore, Michahelles [18] stated in his presentation that RFID technology is currently in the "Trough of Disillusionment" and that for a successful adoption both technology and acceptance (business and consumer) are needed. From the consumer's perspective, the latter will not be given until the privacy issues are solved.

In times of Facebook and YouTube where people upload their personal data, private photos and videos on the Internet to share them with strangers it is hard to find a general definition of the term "Privacy". The OECD (Organisation for Economic Co-operation and Development [46]) defines privacy as follows:

> "It is the status accorded to data which has been agreed upon between the person or organisation furnishing the data and the organisation receiving it and which describes the degree of protection which will be provided."

This definition—even though it has an economic background—implies some important expressions. It implies that no general rules can be defined that will match for all persons in all situations. There needs to be some agreement on which and how data has to be protected and the control has to be kept by the creator or owner of the data. For an automated and non-transparent system like RFID where the user lacks control over the communication flow it must be ensured that no personal data can be accessed without authorization. The following citation was taken from the Directive 95/46/EC [14] of the European Parliament and defines the term "personal data".

> "... shall mean any information relating to an identified or identifiable natural person ('data subject'); an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity"

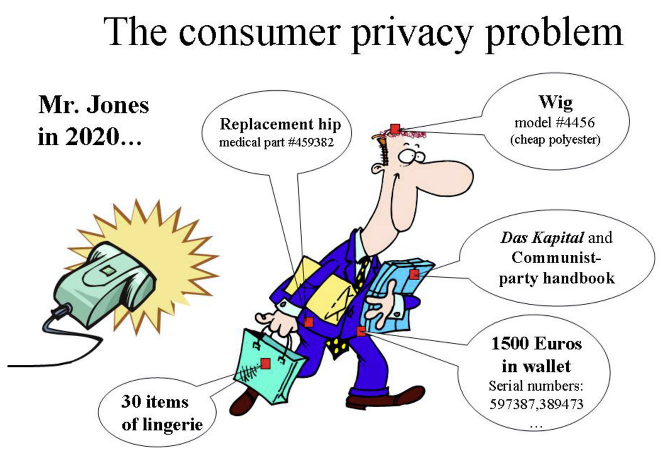

The consumer privacy problem
Mr. Jones in 2020...
Replacement hip medical part #459382
Wig model #4456 (cheap polyester)
Das Kapital and Communist-party handbook
1500 Euros in wallet Serial numbers: 597387,389473 …
30 items of lingerie

Figure 3.5: Consumer Privacy Problem, Figure Provided by Courtesy of Ari Juels [31]

Figure 3.5 is a very famous illustration from Ari Juels work "RFID Security and Privacy: A Research Survey" [31], and shows what could happen if personal data is not protected. The man gets scanned by an RFID reader and all products with attached RFID labels respond with a unique ID. Not only the things themselves give information that can be used to characterize the man in an obviously awkward way, also the uniqueness of the serial numbers can be used to track the man and to study his preferences, consumerism, etcetera. Methods how to provide privacy in RFID systems are explained in the following.

Privacy-Preserving Methods

Security does not automatically imply privacy and the whole privacy problem is not solved by just encrypting the data and using distinguished authentication protocols. A tag that offers its identity directly or indirectly to every reader—even though no other personal data is transmitted—by sending any information that can be used to track the tag is a privacy risk. Basically there are two categories of privacy-provisioning strategies for RFID systems. The first one can be classified as "Physical Methods" and the second one are "Protocol-Based Methods".
Examples for **physical methods** are *shielding* of tags (e.g., Faraday Cage approach [33]) which could be achieved by metal layers that shield the tag against reader fields. This method could be used in electronic passports (see Figure 3.7a) to prohibit reader access when the passport is closed, or in wallets to prohibit any access to RFID-enabled cards that are inside. An akin approach could be made in an active way by using a "Jamming Device" that produces its own radio field to fully disturb the reader communication in the environment—which might not be legal—or just partially by listening to the reader communication and preventing the tags from answering to privacy-critical requests. Another way that is already integrated in the EPC Gen 2 protocol is called *killing* (see Figure 3.7b). Once an RFID tag is out of its working area (e.g. a consumer good that was bought) the RFID functionality is deactivated by putting the transponder into a permanent and irreversible state in which no more tag communication can be performed. This can also be done by physically separating the transponder chip from its antenna or by destroying the chip itself. "Killing or discarding tags enforces consumer privacy effectively, but it

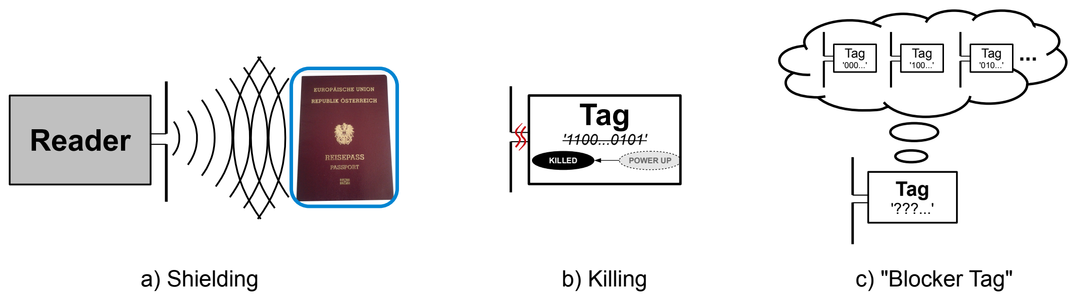


Figure 3.6: Physical Methods for Privacy Preserving

eliminates all of the post-purchase benefits of RFID for the consumer" (Jules [31]). In the same work Jules et al. present a *Blocker-Tag Approach* (see Figure 3.7c) that works for RFID protocols using a so-called "Tree-Walking Singulation Algorithm" for the inventory procedure which is similar to the deterministic inventory that is shown in Figure 2.11. A blocker tag simulates all tags inside a certain UID range and therefore blocks the access to tags inside this range. The work also shows how this approach can be made more "reader-friendly" so that a reader recognizes that a certain UID range is blocked and leaves it out.

Protocol-based methods often involve cryptography in some form (e.g., symmetric-key or public-key cryptography or hash functions) whether on reader or tag side (see Pateriya and Sharma [47] for an overview). Non-cryptographic approaches were presented for example by Nathan Good et al. [21] who suggest not storing any data on the tags that could be used to directly link information to the products (e.g., the ISBN or EPC) or any transactional information. Instead the tag holds some information that can be used as a link to a local database. Furthermore, Good et al. propose the use of random numbers instead of unique identifiers to prevent unauthorized creation of bibliographic directories, but they also state that this does not prohibit tracking. Also Inoue and Yasuura [29] present two approaches to enhance privacy. The first approach uses a public ID stored in a ROM and a private ID stored in a RAM of the tag. The idea is that the public ID can only be read if no private ID was stored. Using a public ID is helpful and wanted inside the production and distribution phase of the product life cycle for tracking purposes. Upon purchasing the product a private ID is stored in the RAM and so the public ID is veiled. This prevents the unique identification of the product on a global domain but keeps this option for a local domain. The second idea suggested in this work, separates the tag's ID into a "Class ID" and a "Pure ID". In analogy to EPC (see Figure 2.19) this would be the *Item Reference* (including the Company Prefix) for the Class-ID part and the Pure-ID part would be the *Serial Number*. By "killing" or rewriting the Class ID the global linkage between the article and the product is destroyed. However, both approaches also do not prevent point-to-point tracking.
Privacy protocols based on cryptography do not automatically imply that the tags need to implement some kind of cryptographic function as Juels et al. showed [32]. In this work the use of RFID tags for banknotes and the created privacy problems are discussed. The underlying assumption is that such tags have to be very cheap which implies that they lack security capabilities. Therefore a public-key cryptographic algorithm is only

implemented on the reader side and the tag does not implement any security algorithm at all. The tag itself contains only the public-key encrypted ID which gets periodically re-encrypted together with a nonce (a random number to vary the cipher text) by a reader. The original ID can only be decrypted by the holder of the private key which is a law-enforcement agency. It is concluded that the privacy level served by this system is not comprehensive.

There are several papers on privacy provisioning by using hash functions. Ha et al. [25] present a protocol for RFID tags that uses a hash function together with a session state for veiling the tag's ID and to prevent an attacker from tracking a tag by connecting him with a previous session (cf. forward privacy). However, Sun and Zhong [55] show how this protocol can be broken by tracking a tag by observing an unsuccessful previous session and present their own hash-based protocol which is explained in the following as an example for a hash-based privacy protocol (see Figure 3.7).

First, the reader generates a random number called r_R that is sent to the tag (1). The tag receives the r_R and generates its own random number r_T (2). Then the tag computes the "hash" (Q) over the tag's ID and the two random numbers r_R and r_T (3), and updates its own ID by calculating the hash value over the ID and writing the result back (4). The Q value is separated into a left and a right side (LT and RT) and the tag transmits the left part of Q together with the calculated random number to the reader (5). Now, the reader knows everything except the tag's ID to do the same calculation of Q as the tag before. Therefore, the reader tries all IDs inside its database (6), and because the tag's ID might already be hashed more often than the ID saved in the database—this happens when the protocol is terminated before the reader updates the tags ID in step (8)—the reader also has to apply the hash function multiple times. The constant "t" defines the upper limit of hash rounds for one ID trial which is described in the paper as the "number of the unsuccessful session runs to be allowed for T" (T is the abbreviation for tag). If the reader found a hashed ID that results in a Q' where the left part matches the LT(Q) that was transmitted by the tag, then the reader transmits the right part of the calculated Q' (7) and updates the database by writing the new hashed ID back. If the reader did not find an ID that delivered the right Q', then the reader terminates the session. The tag then compares the reader's RT(Q') with its RT(Q) and decides if the authentication was successful or not (10).

In that work the authors only estimate the computational and storage costs on the tag side which is justified by following quote.

> "In the RFID system, the tag is usually treated as the resource-constrained electronic device, but the reader is always powerful enough to provide the security service."

It is of course true that the reader does not have such strict constraints but the computational effort of such a protocol on reader side could lead to significant delays and therefore reduce the inventory rate dramatically. Assuming a few thousand tags in the reader database, hashing all of them—in a worst case scenario up to "t" times which was stated by the authors to be "1000 or 10000"—is an enormous effort. This would have to be done, for example, if a not listed tag—which could also be an attacker—sends an LT that matches none of the tag IDs in the reader database.

There exist further works on privacy for RFID systems and similar to security there is not one right way to preserve privacy but many wrong ways. In the end it is a trade-off be-

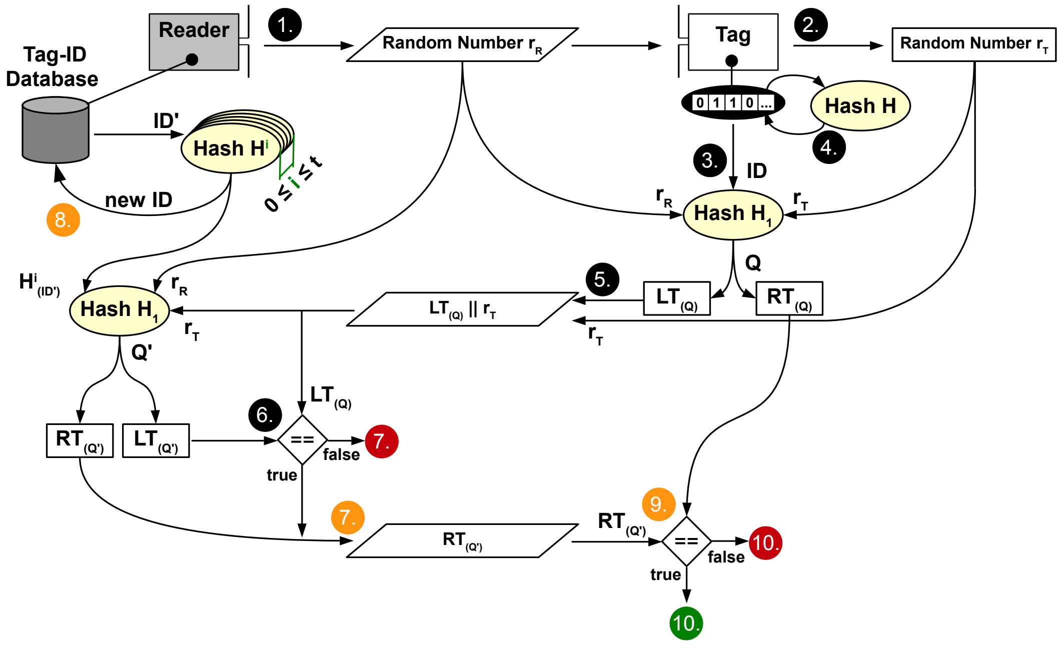


Figure 3.7: Hash-Based Privacy Protocol

tween the privacy level, implementation costs of the tags, infrastructural effort, inventory speed, etcetera.

Chapter 4

Verification and Validation of a Hardware Design

Long before the first integrated circuit (abbreviated IC) was designed, the complexity of electronic circuits was quite low compared to state-of-the-art ICs and consisted only of a few discrete components which could be soldered by hand. The use of increasingly complex electronic devices for military purposes around World War II pushed the development of new techniques which simplified the manufacturing process and lowered the production costs. The invention of the transistor around 1948 was the first step to modern digital technique but it took several years until the transistors replaced the dominant vacuum tubes (see Kilby [37]). The main reason for the low acceptance of the transistor at the beginning was the low transit frequency compared to the vacuum tubes. In 1949 lamination and etching techniques led to printed circuit boards that connected passive components—like resistors, capacitors and coils—and active components (vacuum tubes and transistors) by using conducting paths instead of wires. In the early 1950s Lathrop and Nall developed a photolithographic process to interconnect transistors in a substrate. G.W.A Dummer, who worked for the Royal Radar Establishment, was the first person who realized the potential of integrated circuit technology. The following citation is from Dummer in 1952 on a conference in Washington, D.C. and was taken from Saxena [51].

> "With the advent of the transistor and the work on semi-conductors generally, it now seems possible to envisage electronic equipment in a solid block with no connecting wires. The block may consist of layers of insulating, conducting, rectifying and amplifying materials, the electronic functions being connected directly by cutting out areas of the various layers"

It took six more years until the first integrated circuit was developed. An oscillator that consisted of bipolar transistors was designed by Jack Kilby in 1958 during his work for Texas Instruments. The idea for the realization of an integrated circuit was born when Kilby was alone at the laboratory because he was a new employee and the other laboratory workers went on summer vacation. He was thinking about how to make the circuit production with semiconductors more cost efficient and realized that the passive components could be made out of the same materials as the active components. The conclusion of his thoughts was that it should be possible to build a whole circuit without external wiring for the interconnection of the components. After the vacation he presented his ideas to his chief who supported his work on the IC technology. Kilby was not the only person who worked in this field. In a patent from 1959, Boby Noye the manager of Fairchild R&D showed transistors in a common substrate that consisted of n-regions and p-regions (diffusion regions) that were connected over a metal-layer with an oxide as the insulating material. In the following years, many companies concentrated on the development of new IC technology which allowed to use more and more integrated components to build complex ICs. The rapid development since the first IC in 1958 by Kilby led Gordon E. Moore to the following prediction that Moore presented in an article in 1965 [44].

> "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year... Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer."

This quote is well known as the Moore's Law and is one of the most cited and interpreted quotes in the history of modern technology and still has not lost validity. The time period between doubling of the complexity of ICs was corrected by Moore from one year to two years in 1975 and in the current version 18 months are proposed. The interpretation of Moore's prediction is not only restricted to the complexity of ICs but also on the capacity of computer storage media, the computing power, etcetera. It is obvious that the increasing complexity of ICs also influenced the way of designing electronic circuits. State-of-the-art processors consist of millions or even billions of transistors which is of course unmanageable to be designed, routed and tested by hand. Pure schematic designs, as they were used at the beginning of the electronic circuits, are unthinkable in todays digital design.

The fist hardware description language "ISP" appeared in 1971 and was originally used as a notation in Bells at al. "Computer structures: readings and examples" [4]. The name of the HDL is derived from the intended use for describing the behavior of **i**nstruction **s**et **p**rocessors (abbreviated ISP). ISP was extended to be more suitable as a general descriptive tool for register transfer (abbreviated RT—a term that was also invented by Bell) systems as well as a design tool and for simulation purposes. The synthesis functionality of the ISP-derived tools were limited to special hardware (Digital Equipment Corporations PDP-16 modules) and could not prevail. In the 1980s logic synthesis became an interesting research field for many universities and electronic design automation companies began to adopt the research results to create tools for automatic logic synthesis. At this time, the development of new synthesizeable hardware description languages (abbreviated HDL) became more interesting, since the advantages of an abstract descriptive language to handle complex digital systems were obvious.

The most commonly used hardware description languages nowadays are VHDL and Verilog, whose roots go back to the late 1980s and 1990s, respectively. These languages are the only one that are widely supported by many automatic synthesis tools. The key concepts for both languages are mostly the same and so it is a question of preferences which language is used by a hardware designer (see Table 4.1 and Kaeslin[34] page 177 for details). Another characteristic of both HDLs is that they only support simple functionality to verify hardware modules. As the complexity of the hardware designs increased, the verification functionalities did not longer suffice the requirements of the industry, like reusability, functional coverage, constraint random stimulus generation, etcetera. Hence,

Feature	**VHDL**	**Verilog**
Industry standard	IEEE 1076	IEEE 1364
Initial version	1987	1995
Current revision	2008	2009 (SystemVerilog)
Syntax style	Ada	C
Characteristic	verbose and strict	concise
Logic system	9 states	4 states, 8 strengths
Scoping	yes	no
Strong typing	yes	no
*EDA support	yes	yes

*) Electronic Design Automation

Table 4.1: VHDL and Verilog Comparison

dedicated hardware-verification languages (HVL) like OpenVera and *e* were created to fulfill those requirements. In 2005 Accellera, a consortium of EDA companies, developed hardware-verification features based on the HVL OpenVera and merged them with the Verilog IEEE standard 1364-2005 in 2009. The result was the new SystemVerilog IEEE standard 1800-2009 which combined design and verification in a single language.
This chapter starts with an introduction of a digital VLSI design flow in Section 4.1 to introduce the flow itself and the used terminology which is used in the following sections. Afterwards, different verification metrics that are used to measure the verification progress are then discussed in Section 4.2. Section 4.3 shows an enhanced version of the traditional verification approach based on the scripting language Tcl and compares this approach to a modern SystemVerilog test bench based on the verification methodologies OVM/UVM. Section 4.4 describes the executable specification which was used at the beginning of the project to verify the system-level design and lateron as a reference for the hardware implementation. Finally the laboratory setup is illustrated and described in Section 4.5.

4.1 Introduction of a Digital VLSI Design Flow

All explanations of this Section are related to Figure 4.1 which shows a typical VLSI flow. The different development layers of the digital flow are placed on the left side of the figure. On the right side there are the related tasks and milestones. Of course the flow should not be seen as "waterfall-model" which is worked through layer by layer and task by task. In practice there are cases that need revision of a previous task, e.g., a problem is found in the *Executable Specification* and so the **System Layer** needs to be revised. The time before **tape-out** is called "pre-silicon" and testing in this phase is called "verification". After the tape-out the "post-silicon" phase begins. Testing is then called "validation" because the correct functioning of the chip is validated.
Every project begins with a phase where the tasks, sub-tasks and goals of the project are defined from a behavioral perspective. This is called here the **System Layer** with the related milestone *Specification*. All the desired functionality, characteristics (like power consumption, performance, costs) and operating conditions are defined at the System Layer. Also the project strategy, which parts are designed in-house and which are bought,

hardware/software trade-offs and alternatives are discussed. In order to split the project into sub-tasks, which can be executed independently, data exchange related things like protocols, data formats, interfaces, etcetera need to be set out in the specification.
When the specification is done, a refinement process begins at the **Algorithm Layer** where a collection of suitable algorithms is selected and analyzed to serve the defined requirements. The goal is to find the best solution in terms of computation speed, memory requirements and logic complexity which often is a trade-off. The result is an *Executable Specification* that is used as a proof of concept to verify the specification and serves the system architect as a basement for his work.
A system architect then begins his work on the **Architecture Layer** with a high level description of the future integrated circuit by defining the basic hardware blocks and the interplay of these blocks to implement the selected algorithm. It is the first step from the specification to a hardware implementation in which the system architect decides on the target technology and cell libraries, transfers algorithm functions to a feasible hardware implementation and estimates the hardware costs in terms of chip size, power consumption and costs. The resulting document contains an abstract overview of the circuit design that includes data paths, controllers, analog parts, e.g., level-shifters and voltage regulators, memories and important signals. After the main blocks are defined, the system architect refines the digital blocks and creates a register transfer level (RTL) description of these blocks which contains memory elements (registers) and the computational logic. Further considerations imply clocking, the use of microcontrollers instead of hardwired logic, the memory organization, testing strategies, etcetera. The result is a more detailed description of the previously defined blocks that is written down in the *System Architecture Description* that also contains a floorplan of the integrated circuit.
The **Logic Layer** (also called **Digital Front-End**) contains the description of the digital hardware in a chosen hardware description language (mostly VHDL or Verilog) and on a selected abstraction level (RTL or gate-level). Normally it is not necessary to describe every part of an integrated circuit on gate-level, since the logic synthesis automatically creates a *gate-level netlist* out of the RTL code. In some cases the description on gate-level is necessary when the synthesis of the RTL code produces an awkward gate-level description. After the creation of the *gate-level netlist*, the *Electrical Rule Check*, simulation and verification with timing parameters of the gates can be performed.
On the **Physical Layer** (also called **Digital Back-End**), the step from the gate-level description of the integrated circuit to the actual layout is done. The layout flow includes the placement of the gates and their interconnection (*Place and Route*) based on the *Floorplan*, the *Clock-Tree Generation* to guarantee an even clock distribution in order to fulfill the timing requirements and the extraction of the timing parameters for the gates based on the layout. The digital layout flow is of course almost completely automated. However, the connection of the digital layout to the analog parts and pads is performed by a layout artist. In order to have a high yield of functioning chips after the fabrication process, certain design rules have to be taken into account by the layout artist like the minimum distance of wires or the even distribution of layers on all regions of the chip. The checking of the design rules is done automatically in the *Design Rule Check*. In the *Layout versus Schematic* it is checked that all components that are part of the schematic are used in the layout and all connections between those components are established. The timing information changes because of the created layout which makes it necessary to verify the correct timing of the circuit again.
After the layout is done and the verification procedure did not find anymore bugs or tim-

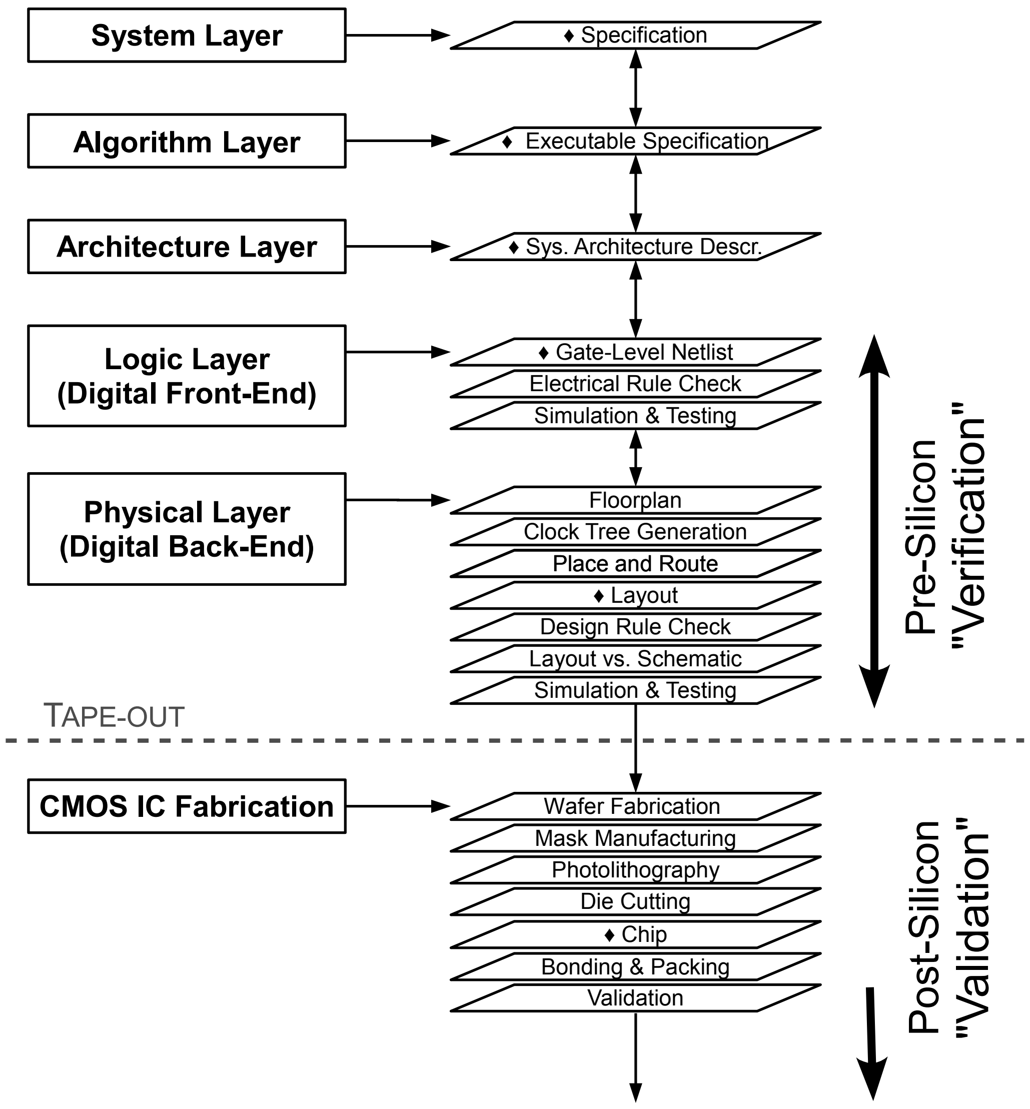


Figure 4.1: VLSI Design Flow with Related Tasks and Milestones

ing problems, the fabrication of the integrated circuit begins by sending the layout to a manufacturer. This step is called **Tape-Out** and is critical since no further changes of the layout—that would influence the ongoing fabrication process—are possible. The fabrication process—that takes several weeks—includes the creation of the masks that are used for the photolithographic process (respectively the creation of the diffusion regions), the production of the wafers, the cutting of the dies and the bonding and packing. The finished chip is then validated in the laboratory under the specified temperature conditions, with different supply voltages within the specified range and other previously defined operating conditions. Post-silicon *validation* is mainly done manually which results in higher costs compared to the pre-silicon *verification*.

4.2 Verification Coverage Metrics

The majority of costs arise after the tape-out which makes a good and efficient *verification* strategy very important, since a bug that is found after tape-out results in higher costs than a bug that is found before. In order to measure the verification progress there are different metrics to measure the coverage of a verification process. The following text gives a short overview about three commonly used verification coverage metrics. For more information see Tasiran et al. [56].

4.2.1 Code-Coverage Metric

A very simple form of a coverage metric is the so-called “Code Coverage” which also appears in software programming languages. Every HDL code consists of declarations, assignments, branches, loops, function calls and others. Branches like “if ... else” or conditional loops are called control-flow instructions and can be represented by a control flow graph where each branch is a possible way trough the code path. To achieve 100 % **branch coverage** each branch needs to be accessed at least once during the simulation. A more accurate way to calculate the code-coverage rate is to check that each existing path of the control-flow graph was accessed at least once. This requires an algorithm that choses all accessible paths through the control flow graph which is not a trivial problem. It is also not necessary—and could lead to an enormous verification effort—to check all existing paths since a sub path might already be part of another path. Therefore a heuristically selected subset of paths—that contains only linearly independent paths—is normally used for the **path coverage**. Besides the two described ways to implement the code-coverage metric there exist other forms like **expression coverage** or **code line coverage** (see Tasiran et al. [56], page 3).
The implementation of the code-coverage metric is quite easy and therefore it is used in many commercial simulation tools. To calculate the coverage rate the number of accessed paths or branches is divided by the number of existing paths or branches and multiplied by 100. The significance for hardware verification is not comparable to sequential software programs since hardware tasks run in parallel. Nevertheless, code coverage metrics are useful and a rate of 100 % should be the goal of every verification engineer but further metrics are needed to guarantee complete functional verification.

4.2.2 Toggle-Coverage Metric

The idea behind the “Toggle Coverage” metric is to look at every point of the circuit that could change during the simulation and check if every state has been reached. Usually every point is seen as a binary node and the states are logical “0” and “1”. By splitting the HDL code into two parts, the data path and the control path, more dedicated metrics can be implemented with different investigation goals. The data-path part of the metric is very useful to check if every register is correctly initialized, set, reset and all register paths are used during the simulation. For the control-path part, the metric represents the communication flow between different parts of the circuit which indicates the use of all circuit features. The toggle-coverage metric in general is suitable to uncover parts of the circuit that are not or not often enough accessed during the verification process. This information is useful for verification engineers to write directed tests to exercise these parts. On the other hand it is not always possible to automatically extract useful informations and to locate the weakness of the tests. Although the insufficiently exercised parts of the circuit could be identified, generating the stimuli that will exercise those parts could be a difficult task.

4.2.3 Functional-Coverage Metric

The verification features of SystemVerilog (SV)—respectively the methodologies OVM and UVM (see 4.3.3)—are strongly related to the “Functional-Coverage” metric. Since SystemVerilog uses constrained random verification instead of a directed testing approach, a metric is needed to check which functionalities have been tested during the simulation. The advantage of functional coverage is that missing functionalities that were specified and implemented in the SV model will be reported. This would not be the case for toggle coverage or code coverage because only the already implemented HDL code is checked. The disadvantage is a notable additional effort for creating a good functional-coverage metric. A very good knowledge of the whole system and a lot of experience is needed to locate the critical parts of the system in order to implement a metric that is suitable to measure the coverage. Ideally, hardware designers and verification engineers work in parallel on the implementation of the previously defined specifications. While the hardware designers implement the digital hardware in an HDL, the verification engineers create a verification plan and implement a model of the system that is used to crosscheck both implementations. The verification plan serves as a basis for the creation of the functional-coverage metric. SV uses so-called “cover points” to define which information should be gathered during the simulation. This information is then used after the simulation to check which cover points were exercised and which are missing. The knowledge about missed cover points is important for the verification engineer to detect weaknesses or possible improvements of the verification environment. When a cover point is never reached even though the test was started many times with random seeds, the verification engineer needs to adjust the constraints for the constrained random verification to pull the tests into the right direction. Although functional coverage seems to be more complex and requires more expertise than other metrics, it is the best way to guarantee high coverage and is indispensable for complex designs. Furthermore, it should be used together with other metrics like code coverage to check that the functional coverage points are sufficient or if more corner cases need to be checked (see Spear [53], page 330).

4.3 Script-Based Verification versus OVM/UVM

Many electronic design automation (EDA) programs support different scripting languages like Tcl (e.g., Cadence, Mentor Graphics, Synopsis etcetera). In many companies Tcl is also used for writing automated functional hardware-design tests. In the following chapter, an example for a typical Tcl test bench is discussed.

4.3.1 An Advanced Tcl Test Bench

Compared to traditional Verilog verification environments, the Tcl-based verification environment in Figure 4.2 works on a higher level of abstraction, which allows to write complex tests with little effort. This results from a paradigm change—time domain to functionality. In a simple Verilog verification environment, it has to be taken care of signal transitions and the time when they occur. The Tcl stimuli file in Figure 4.2 consist

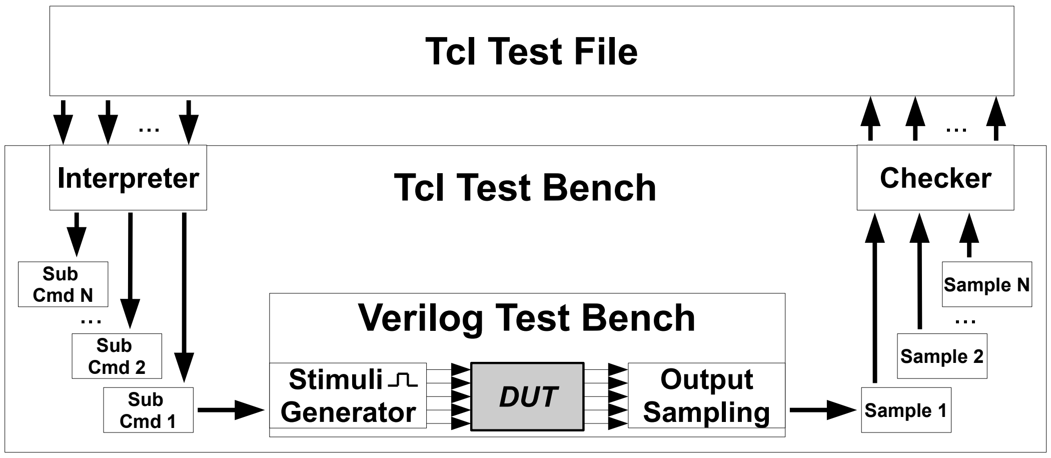


Figure 4.2: Tcl Test Bench

only of high-level commands, which are interpreted by the Tcl test bench and produce one or more sub commands. These sub commands trigger tasks in the Verilog test bench, which are sequences that take care of correct signal transitions. For a person that writes the stimuli files, this means that the time domain has not to be taken into account and the focus is on the DUT's functionality. Also some of the result checking can be handled by the Tcl test bench, e.g., protocol-specific functionality like parity-bit checking. In the Tcl stimuli file the collected information is used to decide if the test was passed or failed. Typically, the Tcl verification environment consists of a set of tests, each directed to a certain device functionality. These so-called directed tests are checked to ensure that the functionality declared in the specification is fulfilled. For larger designs, a verification plan that describes the features and how they are tested, is therefore essential. Using only directed tests to achieve 100 % verification coverage is unsuitable because of the enormous number of directed tests it would need to verify a complex design. It is also impossible for a verification engineer to think of every possible exception that could happen. The solution for this problem is a self-checking test bench that uses constraint random verification, which randomly generates every valid stimuli for a DUT and checks the results automatically using an abstract model of the design. Constraint random verification is one of the main concepts in SystemVerilog which is the basis for methodologies like OVM (open verification methodology) or UVM (universal verification methodology).

4.3.2 A Layered SystemVerilog Test Bench

A typical SystemVerilog environment is shown in Figure 4.3 (for detailed information see Spear[53] page 19). Like the Tcl test bench in Section 4.3.1, this test bench uses different layers to provide a high level of abstraction. On the lowest level of abstraction, the so-called Signal layer, is the design under test (DUT) and its connection to the verification environment. The Command layer consist of a Driver and a Monitor. The Driver converts the input commands into signal transitions and drives it into the DUT. The Monitor works the other way round and takes signal transitions from the DUT's output and groups it to

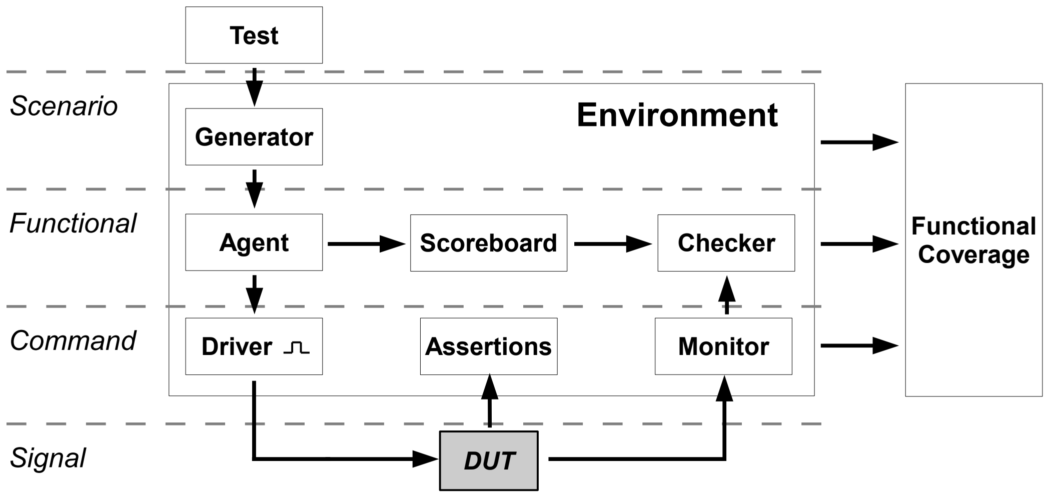


Figure 4.3: SystemVerilog Test Bench, adapted from Spear [53], page 19

commands. Assertions are command specific and therefore also part of the Command layer and check for simple erratic behavior of the DUT. In the Functional layer, the Agent deals with high-level transactions and splits them into simple commands that are then driven by the Driver. The scoreboard also receives these commands and uses a model of the DUT to predict its behavior. Finally, the Checker uses the Scoreboard output and compares it to the transaction observed by the Monitor. The result of the comparison is then used to decide if any further steps are needed, e.g., error logging, trigger functional coverage or immediate stop of the test. The Scenario layer deals with only high-level functionality and uses constraint-random values instead of hard-coded values like in a directed test. A typical scenario for an RFID tag is an Inventory sequence. Hence, constraint-random values are used, the Inventory sequence does not always consist of the same commands and the result therefore also varies.
On top of the test-bench hierarchy is the Test layer that contains the Test and Functional Coverage. These are the only blocks that should be changed during a verification process. Everything else in the Environment should stay unchanged or should only be extended. The Test conducts the functional blocks of the DUT by executing different scenarios and also defines the settings and further constraints for the test run. The Functional-Coverage block is used to control which test goals, defined in the verification plan, have been covered. At the beginning of the verification process, many bugs will be found without setting any further constraints in the Test and the coverage progress will proceed quickly. As soon as the coverage progress got stuck or no further bugs are found, new constraints have to be defined to reach uncovered functionality. At the very end of the verification process, the constraints will be as restrictive as for directed tests to achieve 100 % coverage.

4.3.3 Verification Methodologies

SystemVerilog provides the language basement for many verification methodologies like VMM (verification methodology manual) that was developed by Bergeron et al. [5] originally for the hardware verification language OpenVera. OVM (open verification methodology) and its successor UVM (universal verification methodology) are newer methodologies and were developed for testing complex designs. UVM still uses many concepts and ideas from VMM. UVM/OVM provides a package of SystemVerilog base classes and functionality like reporting, inter-task communication, configuration, advanced OOP (object-oriented programming) techniques like factories or call backs etcetera. Once written, the code can be reused in other projects that use the same component. The main advantage of these methodologies is the standardization of the verification concepts, which is essential for complex designs. Of course this somehow leads to implementation overhead because not every methodology function is used in every test bench, but there are standardized solutions provided for the most common problems a verification engineer has to face when creating a functional hardware verification environment. An OVM/UVM verification environment looks very similar to the SystemVerilog environment shown in Figure 4.3, except some terms differ or components like the Agent encapsulate more functionality. Hence, UVM is the newest of the verification methodologies, the following section will describe the UVM library and its features.

4.3.4 The Universal Verification Methodology's Library

The UVM library comes with a bunch of classes which are used to implement the verification environment (see Accellera UVM Class Reference Manual[1]). Every UVM data or hierarchical class is derived from the "uvm_object" class (see Figure 4.4) and must implement its pure virtual methods. The uvm_object class contains basic functionality for naming, creating, (un)packing, comparing etcetera.
Reporting is used to handle messages that are produced by UVM components. Every uvm_report_object and its deductions provide an interface to a central report facility. This facility is used to filter messages and trigger actions based on the report message arguments. Report-message arguments consit of a unique id, a message text and optionally a file name, a line number and the verbosity. The verbosity argument is a number that the report facility uses to decide whether to issue a message or not, based on the configuration of the report facility. When a report message is issued, then the action settings of the uvm_report_object decides how to handle the message. Such actions can be printing of the message, logging to a file, quitting the simulation etcetera and these actions can also be combined.
Components like agents, sequencers, drivers or monitors are the hierarchical elements that form the verification environment (Figure 4.5). Each component can be accessed by its hierarchical path or through search methods. Configuration allows to set component parameters, e.g., to change its behavior in a new test run. Another feature of the components are phase hooks, which are triggered when a certain verification phase is entered and executes component-specific code. Phasing is one of the central UVM principles which partitions the verification process into different phases. UVM uses factories for component instantiation, which is a powerful OOP design pattern to dynamically generate or override objects. The main advantage of a factory construct is that an object that calls the factory to create another object does not need to know the concrete implementation of the object which should be created. It is enough to know the interface or the base class of the object. This allows to change or extend components without the need that other components have to take care which deduction of the base class is used. An example for

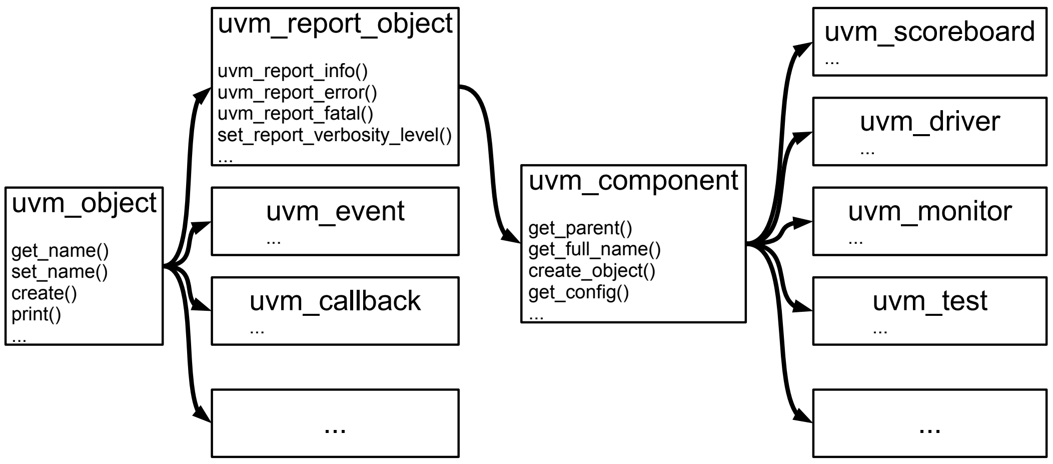


Figure 4.4: Partial UVM Class Overview

the use of the factory is a verification environment that uses a model of an EPC Gen 2 tag (derived from a uvm_component) to verify the monitor observations. To verify optional and specific EPC Gen 2 commands, like cryptographic commands, the standard model in the factory is overridden in a uvm_test file with an extend model without changing the other components.
Since every component runs in an own task, inter-task communication is needed. In UVM, components are connected trough transaction-level modeling (TLM) interfaces of the library, so-called ports. **TLM ports** provide different methods, either blocking or non-blocking, for data exchange in a mailbox-like way. The advantages of TLM ports are the reusability due to non-hierarchically fixed connections, they are easy to use and provide a high level of abstraction.
One difference to the SystemVerilog environment in Figure 4.3 is, that the UVM environment has no component named “Generator”. In UVM the generator is called **sequencer** and is part of an active agent, which also contains the monitor and a driver. Passive agents only contain a monitor component. The job of a sequencer is to execute one or more sequences in parallel, selected by a uvm_test, and to guide the driver to set the correct DUT signals. On the top level of the test-bench hierarchy is the uvm_test class. The instantiation of the uvm_test class sets the configuration of the components, chooses the sequences and defines further constraints for the randomization. At the beginning of the test phase the test object may contain only the instantiation of the environment and a default configuration of its components. Even by exhaustive testing, with wide constraints, some functions of the DUT may never be touched. This can be discovered in the functional coverage report at the end of the test run. To pull the test run into a certain direction, i.e., to cover every exceptional case, the test class is the right place to create such forces.

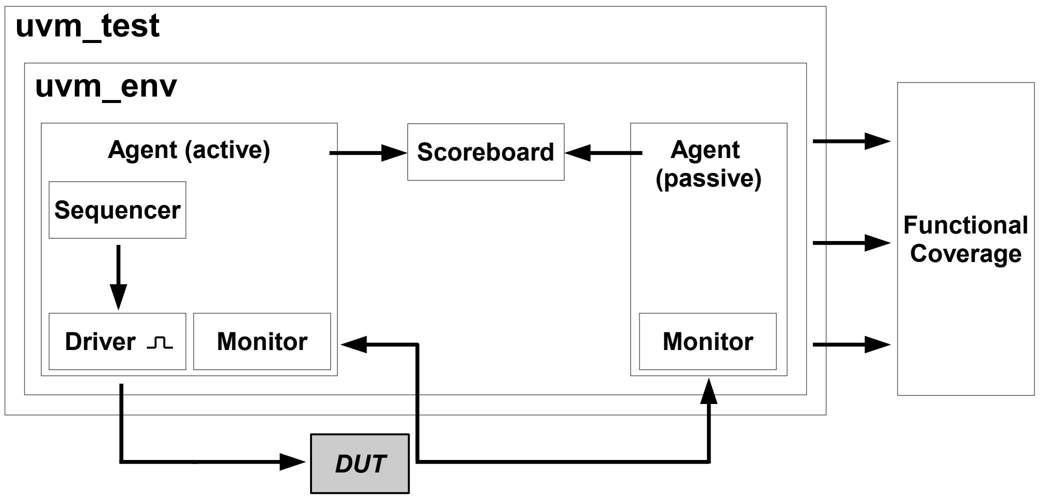


Figure 4.5: UVM Test Bench

4.4 Executable Specification

The most important step in a hardware design is the specification, since it is the step on which the actual implementation and the verification plan etcetera base on. A mistake in the specification step that is found during implementation or lateron could lead to an enormous cost overhead caused by redesigning of the system. The most serious mistakes are conceptual flaws, which can easily be avoided by simulating the system in a virtual environment, a so-called Executable Specification. The specified system and its communication to its environment is simulated on a high level of abstraction. By using high-level programming languages like C#, an abstract model of the system can easily be generated with the comfort of object-oriented concepts. This model is used to prove the concepts defined in the specification. It can also be a advantageous during implementation, to compare the behavior of the model and the implementation.
The Executable Specification that was used in this project, contains the classes shown in Figure 4.6. Here, the "SecuredDevice" is the simulated system and the "BackendSystem", the "Reader" and the "Microcontroller" form its environment. The "BackendSystem" generates the symmetric keys for the "SecuredDevices" which are derived from its master key and the EPC of the tag. In order to use the security-enhanced functionality of the "SecuredDevice", the "Reader" needs to know this key. To receive the key of a certain tag, a secure communication channel between "Reader" and "BackendSystem" needs to be established. This is done by using a public-key method like RSA (or ECC) for mutual authentication and by exchanging a session key. The session key is then used to secure the further communication using a symmetric-key method like AES, e.g., to exchange the key of the "SecuredDevice". The "Reader" knows different EPC Gen 2 commands including the "Authenticate" command, which is device specific and can be used to implement different authentication methods. The "Microcontroller", which is part of the laboratory setup and communicates over a wired interface with the "SecuredDevice", is not simulated. Main component of the Executable Specification is the "SecuredDevice", consisting of an EPC Gen 2 tag, an AES core and the wired interfaces. The EPC Gen 2

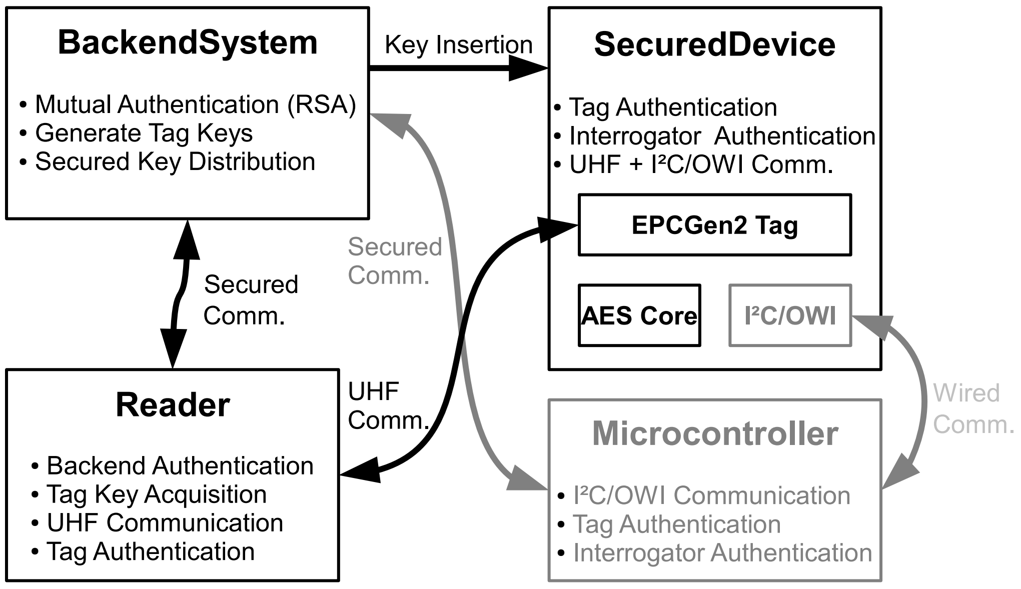
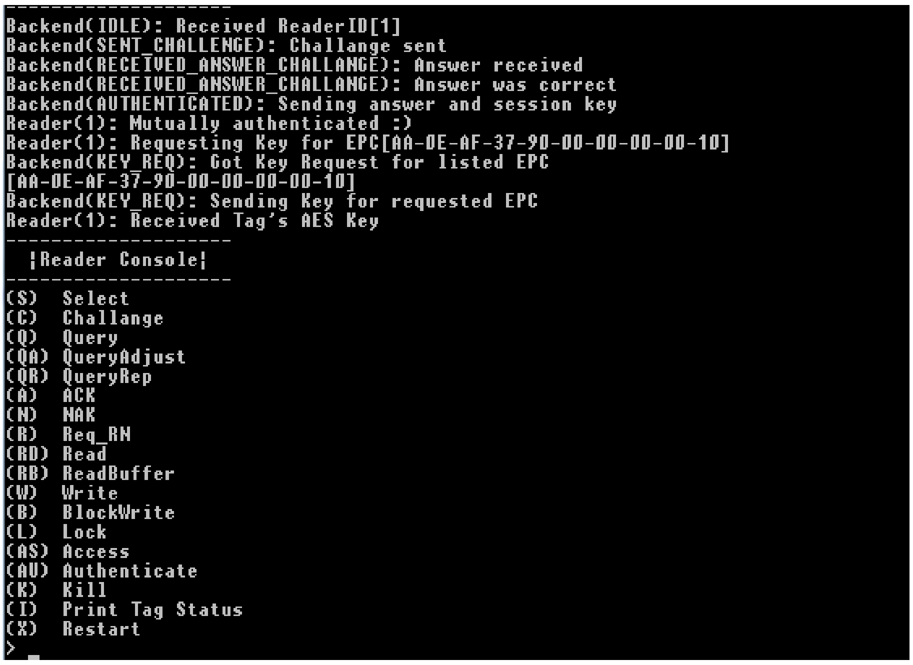


Figure 4.6: Executable Specification Classes Overview

tag understands all mandatory commands and reacts according to the standard. Figure 4.7 shows the console output of the Executable Specification on start-up. The "Reader" to "BackendSystem" communication is done automatically. UHF commands can be send interactively by typing commands into the console. The communication flow on message transfer level is printed in the console. Furthermore, the state of the "SecuredDevices" can be printed, containing all relevant information to control its behaviour.

4.5 Validation and the Laboratory Setup

Validation of an integrated circuit is a task that is not completely automated like the verification process and usually consists of several steps that need to be done by hand. The validation strategy is part of the system-architect description and needs to be implemented and verified like the other integrated-circuit functionalities. There are different verification strategies (see Söser [52], page 133).
The standard verification strategy is the **external verification**, which involves an external test hardware to check the correct functionality of the chip. Therefore, the external hardware generates a test pattern and checks the behavior of the circuit and returns the result. The generation of the test patterns can either be deterministic, random or pseudo exhaustive. Deterministic validation implies the selection of the test patterns by hand, e.g., a test script. For the pseudo-exhaustive strategy an additional error simulation model is needed that generates test patterns that can be used to check for a specific error.
The **specialized-hardware** validation strategy can only be used for a small amount of manufactured chips. Typically the chips are integrated into the destination hardware and



```
--------------------
Backend(IDLE): Received ReaderID[1]
Backend(SENT_CHALLENGE): Challange sent
Backend(RECEIVED_ANSWER_CHALLANGE): Answer received
Backend(RECEIVED_ANSWER_CHALLANGE): Answer was correct
Backend(AUTHENTICATED): Sending answer and session key
Reader(1): Mutually authenticated :)
Reader(1): Requesting Key for EPC[AA-0E-AF-37-90-00-00-00-00-10]
Backend(KEY_REQ): Got Key Request for listed EPC
[AA-0E-AF-37-90-00-00-00-00-10]
Backend(KEY_REQ): Sending Key for requested EPC
Reader(1): Received Tag's AES Key
--------------------
  |Reader Console|
--------------------
(S)  Select
(C)  Challange
(Q)  Query
(QA) QueryAdjust
(QR) QueryRep
(A)  ACK
(N)  NAK
(R)  Req_RN
(RD) Read
(RB) ReadBuffer
(W)  Write
(B)  BlockWrite
(L)  Lock
(AS) Access
(AU) Authenticate
(K)  Kill
(I)  Print Tag Status
(X)  Restart
>
```

Figure 4.7: Executabel Specification Start-Up Console Output

then verified. Additional costs that might occur when a chip is not correctly working must be taken into account.
Another validation strategy is called **built-in self test** (abbreviated BIST) which is a validation procedure that is either triggered automatically when the chip is powered and the BIST functionality was not deactivated, or is triggered externally. Like for the *external verification* there exist different ways to generate and select test patterns that are applied to the system. The difference is, that the generation of test patterns, the sequential control of the verification process and the result checking is done inside the chip.
Prototype validation is done before the chip is released for mass production. Besides the validation of the chip functionality, the temperature behavior, the behavior when the voltage supply is lower or higher then the normal supply voltage, the power consumption, etcetera are characterized. When errors are found during validation, the error source can be found by using laboratory equipment like different kinds of microscopes, structural analysis of the chip with lasers or X-rays, ion beams or by probing of the chip signals with fine needles (only possible when test pads are available for the signals).
Besides the validation that is done after the chip was manufactured there are tests that are performed to monitor the manufacturing process. Thus, for example, the scribe border—which is the line that is used to cut out the dies—is used to generate monitoring structures of every used process layer that allows optical assessment of the manufacturing process. Early detection of errors is necessary to save costs because the later the error is detected the more unnecessary manufacturing steps are performed. The course of this cost factor that is produced at different detection times during the production is shown in Figure 4.8. It is assumed—according to Söser [52] on page 132—that the costs for an undetected error increase by a factor of 10 during the listed production steps.
For the validation of the EPC Gen 2 chip a laboratory setup like in Figure 4.9 is used. The chip ("Secured Tag") is connected to the verification environment over four pins. Two

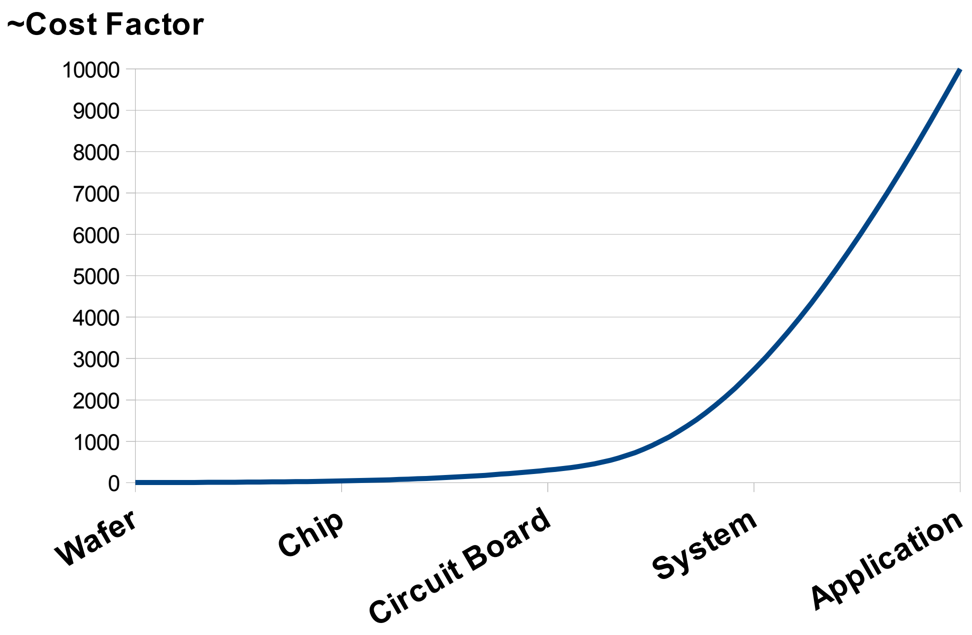


Figure 4.8: Cost Factor Course of an Undetected Error at Different Production Times

pins are used for the wired interfaces I^{2}C and one pin for OWI, respectively. Depending on the interface that is used, a different wiring with passive components like pull-up resistors, capacitors or diodes are needed for the *SCL* and *SDA* pins. The other pins are the antenna-connector pins that are connected to the signals that carry the signal ground and the carrier with the modulated subcarrier. To demodulate the backscattered answer of the tag, a low-pass filter is applied to the antenna signal before it is sampled by the data acquisition (abbreviated DAQ) card. The validation process is controlled by a computer that runs a validation script. The script generates the output signals of the DAQ card and checks the correctness of the tag answers. An example for a validation script is shown in code listing 4.1. At the beginning of the script the radio signals are disabled and the I^{2}C interface is configured. The actual communication is done by the "I2C_Write" and "I2C_Read" commands. With the given "pass" argument the script interpreter expects that the tag will response correctly and will report any misbehavior at protocol level. Further arguments are the I^{2}C slave address of the tag, the destination or source address and the data. For the "I2C_read" command an optional length and expected answer pair can be set, which is checked for correctness when the "pass" argument was set. Besides the protocol-specific commands, there are a couple of commands to control the validation process, e.g., the "SYS_Wait" command that waits for the stated number of milliseconds.

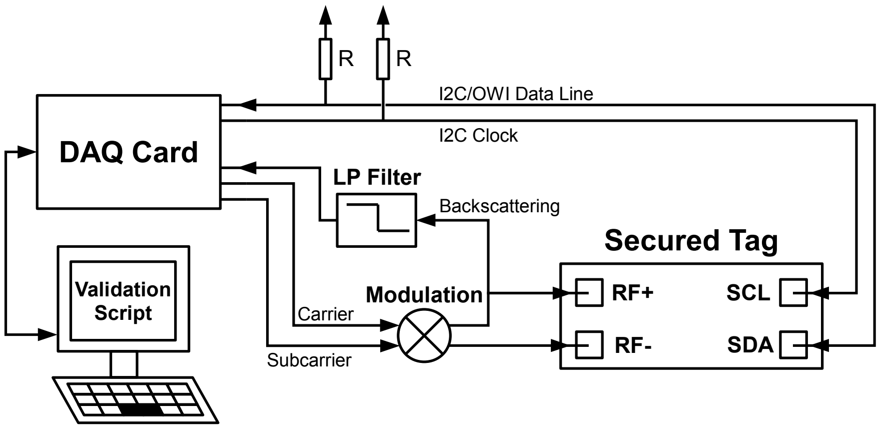


Figure 4.9: Laboratory Setup for Validation of the Tag

Listing 4.1: Validation Script Example

```
# Initialize
SYS_RF off     # Disable RF
SYS_Power 0

SYS_VDD on # Enable I2C @ 400 kHz
SYS_I2C on
SYS_I2C_Frequency 400

# 1. Enable AES core regulator
I2C_Write pass D1 0000 01

# 2. Reset the AES Core
I2C_Write pass D1 0000 02

# 3. Start the interrogator authentication
I2C_Write pass D1 0000 03

# 4. Wait until challenge was created
SYS_Wait 1

# 5. Read challenge data
I2C_Read pass D1 0030 16 00000000000000000000000000000000

...
```

Chapter 5

Practical Implementation of an EPC Gen 2 Tag

The goal of the practical part of this work was to design and implement a security enhanced passive EPC Gen 2 tag, which combines both contactless and contact-based authentication methods. Besides the EPC Gen 2 UHF interface, the tag should also have two wired interfaces (I^2C and a One-Wire Interface, see Section 5.1.5). For the cryptographic operations an AES core was used. Starting point was a complete hardware implementation of an EPC Gen 2 tag, including the wired interfaces, an EEPROM memory, etcetera and a verification environment. The main part of the practical work was to integrate the AES core, connect it to the tag interfaces and implement specific cryptographic commands. At the beginning a virtual authentication system was designed, including the EPC Gen 2 tag, a reader device and a trust-provisioning backend system. The system was simulated in a so-called "Executable Specification" using the high-level programming language C# (see Section 4.4). After the implementation of the tag, the system was verified using Tcl script tests and the SystemVerilog based OVM (Open Verification Methodology, Section 4.3.3). Finally, the layout of the EPC Gen 2 tag was created, verified with simulations and a test chip was produced. In the following, an overview of the tag design is given in Section 5.1 and implementation-related problems are discussed in Section 5.2.

5.1 Design Overview

Figure 5.1 shows the system-level view of the tag design that contains the digital and the analog top modules, the AES core, the EEPROM and the UHF and wired interfaces. All components are connected to the digital top module, which controls the data flow between them. Since every component in Figure 5.1 has its own voltage domain, the analog part of the system contains voltage regulators, charge pumps, voltage-level shifters and comparators that notify the digital top that the components are ready to use or not. The voltage supply is established either via the electric field of the UHF interface (passive RFID technology) or the wired interfaces. Access to the AES core or the EEPROM over the external interfaces is performed indirectly by the AES and memory-control units of the digital part of the system. The controllers regulate access rights, locks, activate voltage supply on demand and convert the external request into signals that can be handled by the components. In the following sections the components and interfaces of the EPC Gen 2 tag will be discussed in more detail. The interaction of the components will be described

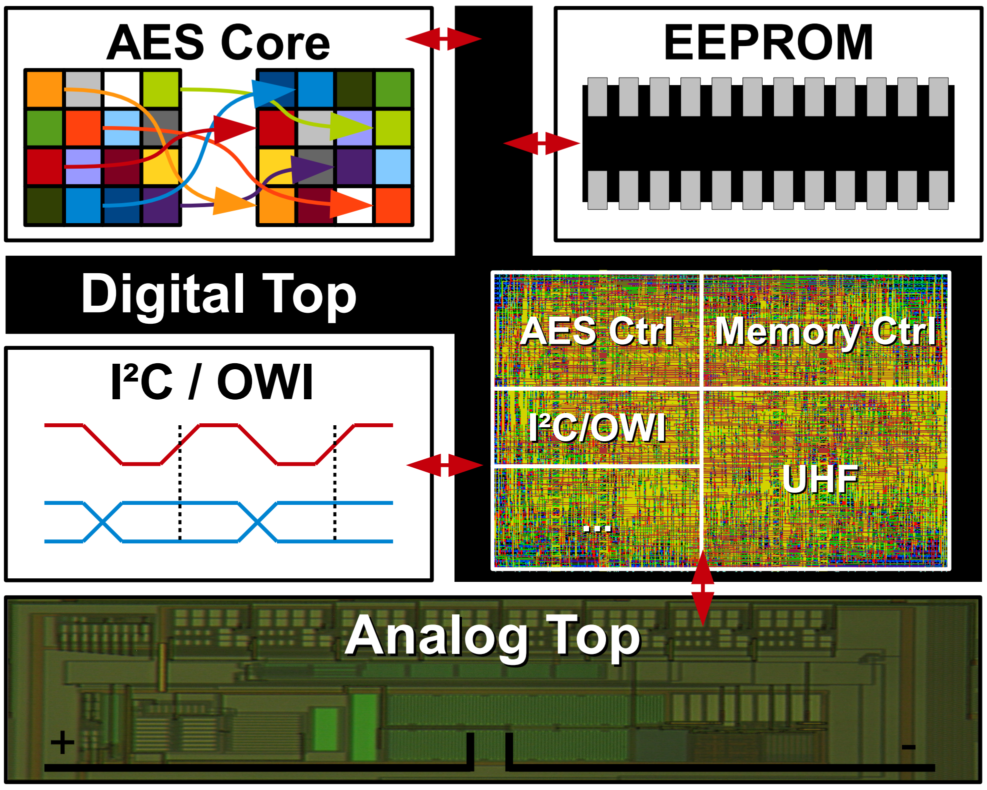


Figure 5.1: System-Level Overview of the EPC Gen 2 Tag Implementation

and the functionality and underling concepts explained.

5.1.1 AES Core

The AES core consumes about 1/16 of the chip area and has its own voltage regulator. When a cryptographic function is requested, the regulator is turned on and a comparator checks if the voltage supply is stable. The access to the AES core is established via a simple bus system where the core acts as a slave device and receives control signals from outside. All communications, including data exchange, triggering functions or reading status information, are accomplished memory mapped (see Figure 5.2) by reading or writing the according registers. Apart the cryptographic functionality, the AES core implements a series of countermeasures against implementation attacks.

5.1.2 AES Controller

The access to the AES core over the UHF or the I^2C/OWI interface is handled by the AES Controller. The AES Controller is connected to the AES core as shown in Figure 5.3. Because the I^2C/OWI interface and the AES Controller run in different clock domains, a synchronization of the control signals is necessary (see synchronization problem in Section

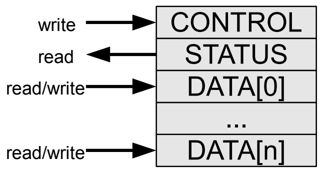


Figure 5.2: AES Core Simplified Memory Map

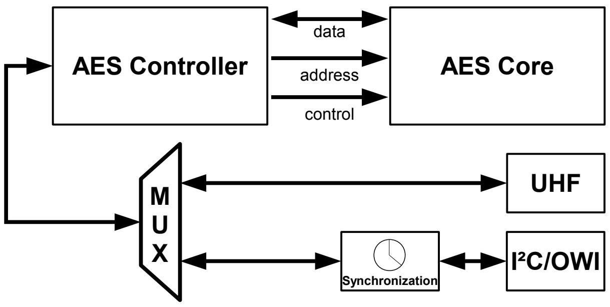


Figure 5.3: AES-Core Communication Paths

5.2.1). To prevent parallel access from different components of the chip to the AES core, the control signals are multiplexed on transaction basis following the principle of "first come, first served".The AES Controller then checks the signals and generates valid bus signals for the AES core. Furthermore, a series of additional functionalities such as address mapping from/to I^{2}C, clock switching and clock generation for the AES core etcetera are implemented. Main part of the AES Controller is the finite-state machine that handles different kinds of authentication services (see Figure 5.4). The authentication services are explained in the following.

I^{2}C Tag-Authentication Method

In order to authenticate the tag, an interrogator calculates a randomized challenge and sends it via I^{2}C to the tag by writing to the AES Data registers (see Figure 5.2). To start the response calculation, the interrogator writes to the AES Control register and triggers the encryption functionality of the tag. The finite-state machine of the AES Controller then transits from *IDLE* to *LOAD_CONFIG*. In this state the AES Controller loads the configuration settings from the EEPROM and configures the AES Core. When the configuration procedure is done, the state transits to *AES_START*. The state change activates the internal clock generation and safely switches the clock source (see implementation problem in Section 5.2.2) of the AES Controller. The finite-state machine stays in the *AES_BUSY* state as long as the AES core has not finished the encryption. After the calculation is finished, the clock source gets disabled again and the AES Controller takes

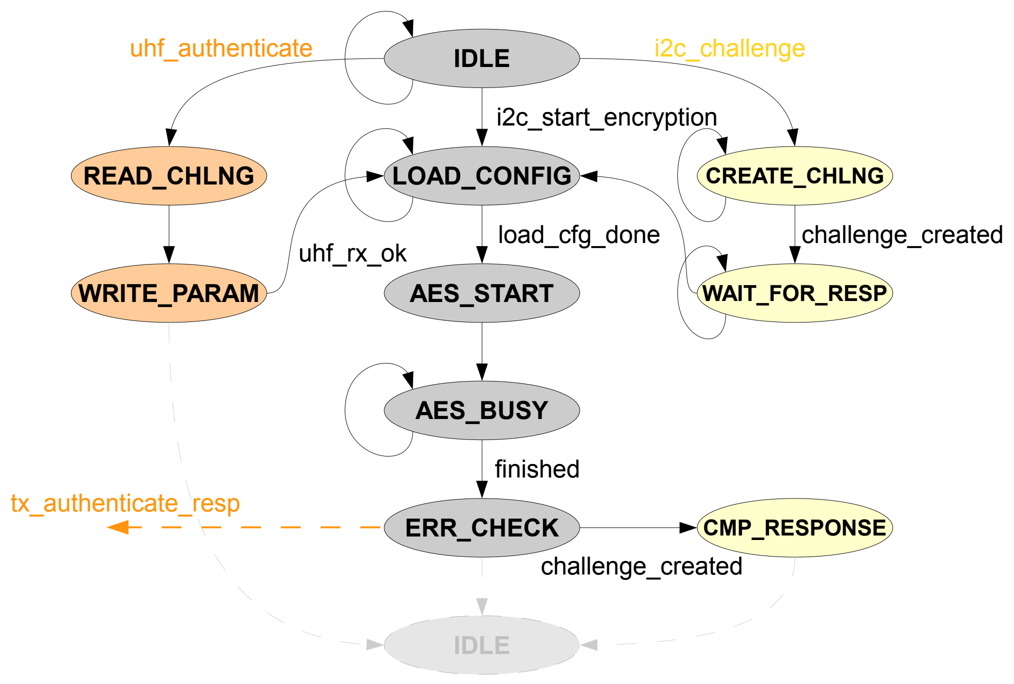


Figure 5.4: AES-Controller State Machine

over the control of the AES core. After an error check the state transits back to *IDLE*. During the whole procedure the interrogator can check the status of the encryption by reading the AES Status register. Finally the interrogator can read the encrypted data from the AES Data register and decrypt it to proof the tag's authenticity.

I^{2}C Interrogator-Authentication Method

An interrogator authentication is triggered by an I^{2}C write to the AES Control register. This transfers the AES Controller state from *IDLE* to *CREATE_CHLNG*, where the internal random-number generator is used to calculate a challenge for the interrogator. When the challenge was created the AES Controller waits until the interrogator reads the challenge, calculates the response and writes the result back. Then the AES Controller decrypts the response—which actually is an AES encryption (see I^{2}C Tag-Authentication Method 5.1.2)—and compares the result to the stored challenge. When the challenge and the response match, the tag enters the *SECURED* state.

UHF Tag-Authentication Method

The EPC Gen 2 V2.0 standard defines an "Authenticate" command frame that can be used by tag manufacturers to implement their own authentication methods and to integrate them in a so-called Cryptographic Suite (CS). The standard only defines a skeleton frame with fields that are supposed to select a CS by its identifier, one that defines how the response is handled and a length parameter. Everything else—how the authentication is done and what state transitions are caused by the authentication result—is defined by

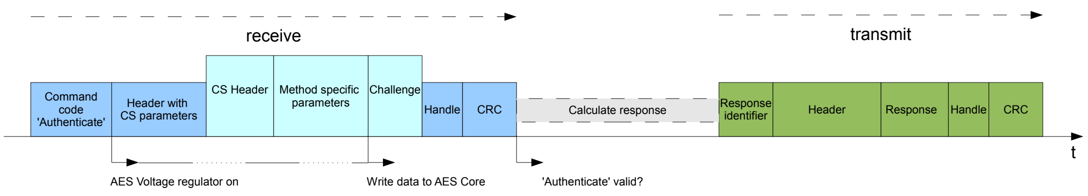


Figure 5.5: UHF “Authenticate” Timing Diagram

the CS and its selected method. Figure 5.5 shows a timing diagram where a reader sends a valid “Authenticate” command to the tag. When the tag receives the command code, the AES-core voltage regulator is switched on. It takes a few microseconds until the AES core is powered and can be used. The “Authenticate” header selects the AES CS that identifies the following embedded frame. Parameters like the authentication method, key-material selection, the challenge data etcetera are defined inside this frame. Hence, the AES voltage supply is stable immediately before the challenge is received, the challenge data can be written to the AES core without buffering the data. The creation of the bus signals for the AES core to write the challenge is done in the *READ_CHLNG* state of the AES Controller. After receiving the challenge some additional parameters like a method identifier and a random number (cf. “salting”) are added. Finally, the tag receives and checks the tag identifying handle and a CRC checksum that are again part of the “Authenticate” command. When all received parameters are valid, the UHF Controller signals the AES Controller to start the encryption procedure (see I^{2}C Tag-Authentication Method 5.1.2). The result of the encryption (or an error code) is then packed into a UHF frame and transmitted to the reader. To proof the authenticity of the tag, the reader decrypts the tag response and compares the result with the original challenge.

5.1.3 EEPROM

The electrically erasable programmable read-only memory (abbreviated EEPROM) is used in many tags to store information that must not be deleted even though the tag is not constantly supplied with power. In this context, this is also referred to as non-volatile memory. The EEPROM that is used for this tag has a storage capacity of 4096 bits and is separated—according to the EPC Gen 2 V2.0 standard—into four memory banks. Besides the EPC Gen 2 memory banks - that store the Electronic Product Code (EPC), the ISO/IEC 15963 allocation class identifier [13], kill and access passwords, etcetera - the EEPROM also stores the tag configuration and implementation-specific information. The following description of the working principles of an EEPROM refers to Figure 5.6 which was adapted from Kuo et al. [39] page 314 and Gawlik [20]. An EEPROM memory cell consists of a memory transistor and an upstream access transistor. The access transistor selects the memory cell for reading or writing and is needed to prevent an unwanted connection between data line and signal ground. This would occur if the memory transistor is self-conducting, due to an uncharged floating gate after a logic “1” was written. The memory transistor saves a logic symbol by charging or discharging its floating gate. When the floating gate is charged a logic “0” is stored, an uncharged gate represents a logic “1”.

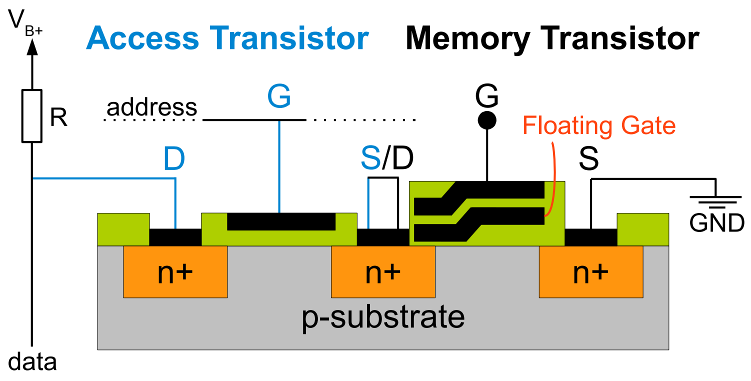


Figure 5.6: Cross Section of an EEPROM Memory Cell

Writing Procedure

Condition: $V_{B+} = 0\,V$, $V_{Gacc} = 12.5\,V$, $V_{Gmem} = 0\,V$, data = "0" (0 V) or "1" (12.5 V)

In the erased state of the memory cell the floating gate is charged ("0") and the memory transistor blocks. The access transistor conducts and connects the drain of the memory transistor with the data line. When 0 V are applied to the data line the floating gate stays charged. For a logic "1", 12.5 V are applied to the drain of the memory transistor. This discharges the floating gate because of an electron breakthrough (Fowler-Nordheim tunnelling [41]) caused by the strong electric field between the gate and the drain terminals. The electrons tunnel trough the (only a few nanometers thick) SiO2 layer between the drain and the floating gate. The threshold voltage of the memory transistor is now negative and the transistor therefore self-conducting. A write cycle requires about 3-5 milliseconds.

Reading Procedure

Condition: $V_{B+} = 5\,V$, $V_{Gacc} = 5\,V$, $V_{Gmem} = 5\,V$, data = ?

For the reading procedure the access transistor is conducting and 5 V are applied to the gate of the memory transistor. If the floating gate is charged, the memory transistor blocks and the data line is pulled to 5 V (pull-up resistor). In the uncharged state, the memory transistor conducts and connects the data line with signal ground. Reading is much faster than writing and takes only a few nanoseconds.

Erasing Procedure

Condition: $V_{B+} = 5\,V$, $V_{Gacc} = 20\,V$, $V_{Gmem} = 20\,V$, data = 0 V

To charge the floating gate very high voltages—in terms of low-power digital integrated circuits—are needed. The access transistor connects the memory transistor's drain to signal ground. Since a high electric field is applied between the gate and the drain, electrons can tunnel trough the insulator and charge the floating gate. The memory cell is now in its default state (erased) and the memory transistor is blocking due to the increased threshold voltage. An EEPROM memory has only limited erasing (writing) cycles (1,000 - 100,000) and each cycle consumes typically about 5 milliseconds.

5.1.4 Memory Controller

The Memory Controller is a comfortable interface from the digital top module to the individual memory cells, each representing a single EEPROM bit (Figure 5.6). As described in Section 5.1.3, depending on the type of EEPROM access, the memory-cell internal transistors need a certain wiring in order to execute a functionality. The setting of the correct signals is the main task of the Memory Controller. Over the interface, the digital top module selects a memory address, the access mode (read or write), data input etcetera. Then the Memory Controller checks the selected address for validity and maps the address to a physical EEPROM address. Furthermore, it is checked if the voltage levels for the memory access are stable and if the memory is not locked. To exclusively pick a certain memory cell, the address is decoded into a row and a line signal (shown in Figure 5.7). The memory-cell transistor terminals are then wired according to the selected functionality. For a read access, the active memory cell forces the *bit_io* line to signal ground if a "0" bit is read. When a logic "1" was stored before, the memory cell enters a high-impedance state which pulls the line up to the maximum positive voltage level over a pull-up resistor. The same bit line is also used for writing a bit to the EEPROM. Since the memory cell works with other voltage levels than the Memory Controller, level shifters are needed to convert between the two voltage-level domains. Besides the level shifters, other analog parts like voltage regulators or charge pumps to generate the high voltages that are needed for the memory cells to work correctly are used. Furthermore, comparators are needed to signal the Memory Controller that an EEPROM functionality is available or not.

5.1.5 I^{2}C and One-Wire Interface

Besides the UHF interface, the implemented EPC Gen 2 tag has two wired interfaces to communicate with external devices and to establish access to the tag functionality and the EEPROM. The I^{2}C interface is — according to its specification[48] — "a simple bidirectional 2-wire bus for efficient inter-IC control". The name is derived from its intended use to **i**nterconnect **i**ntegrated **c**ircuits (IIC - I^{2}C). For a bidirectional connection with an 8-bit data bus, the transfer rate is between 100 kbit/s (in Standard mode) and 3.4 Mbit/s (in High-speed mode). With the so-called Ultra-Fast mode, an unidirectional connection with up to 5 Mbit/s data-transfer rate can be achieved. A simple I^{2}C connection contains at least a master device and one or more slaves (see Figure 5.8). For complex connections with more than one master, an additional bus arbitration and collision detection is needed. Each slave has a unique bus address by which the slave is selected at the beginning of a data transfer. The two communication wires are called SDA (data line) and SCL (clock line). Both wires are connected to a pull-up resistor that pulls the lines high when no device forces the line to the signal ground. Only masters can force the SCL line. Theoretically, an unlimited amount of slaves could be connected to one master, but because of the increasing capacitance — when a slave is added — this would lead to a slower transfer rate. At the moment the I^{2}C bus allows only 7 or 10 (less used) address bits. Figure 5.9 shows the two signal lines from the beginning to the end of an I^{2}C transaction. Every transaction begins with the START condition where the master forces SDA to signal

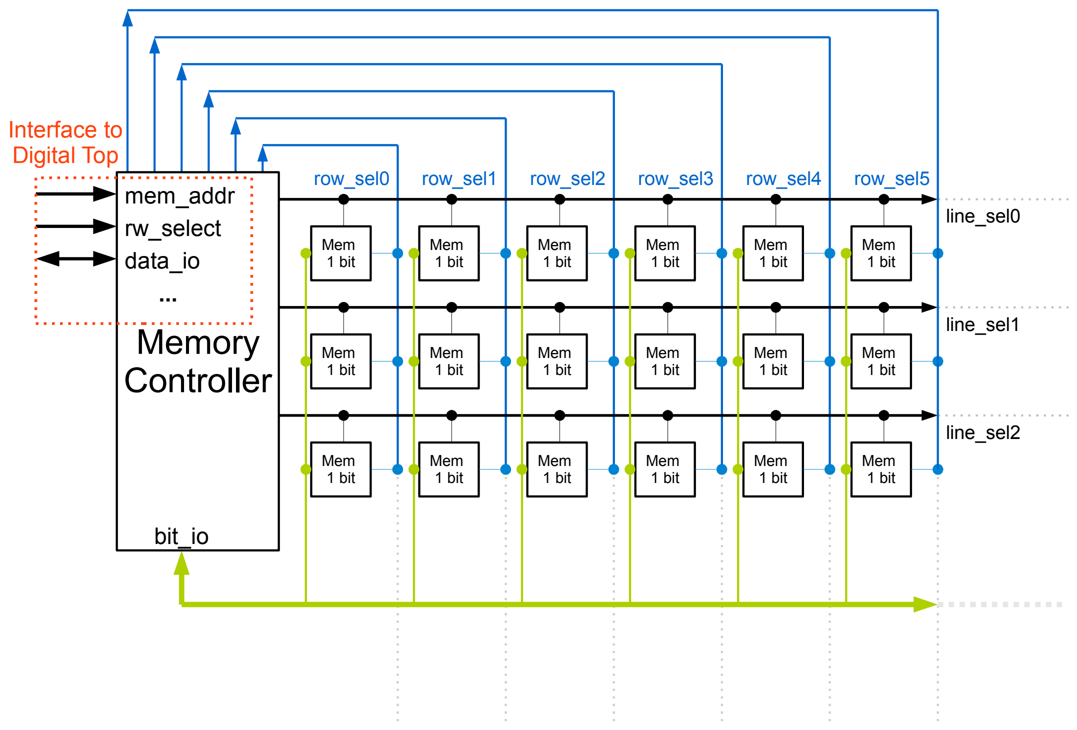


Figure 5.7: Memory Controller with Connection to the EEPROM Cells

ground while SCL is still high. During further communication, the clock remains under the control of the master but the SDA line is shared. Regardless of whether the master or the slave controls the SDA line, it has to be stable during the clock is high. The STOP condition signals that a transaction is complete, which is defined as an SDA signal transfer from low to high while the clock remains high.
After the master starts a transaction by performing a START condition (Figure 5.10), one slave address is transmitted followed by a bit that selects a read (high) or write (low) transaction. For the ninth bit (ACK) the master releases the SDA line. The slave that is associated with the transmitted address acknowledges the request with a low SDA signal. Depending on the transaction type (read or write), either the master or the slave starts sending data and the device has to acknowledge every transmitted byte until the STOP condition is sent.

The **O**ne-**W**ire **I**nterface (OWI) uses only one wire to establish a bidirectional communication with data rates between 1 kbit/s up to 125 kbit/s. The same line is also used for external power supply of the tag over a capacitor, which is either part of the chip or is connected from the outside (Figure 5.11). Furthermore, a diode is needed to ensure that the energy flow is unidirectional and the capacitor does not feed the energy back over the communication wire. Compared to the I^{2}C protocol, the OWI differs only in the representation of the logic symbols and the START/STOP conditions. The communication

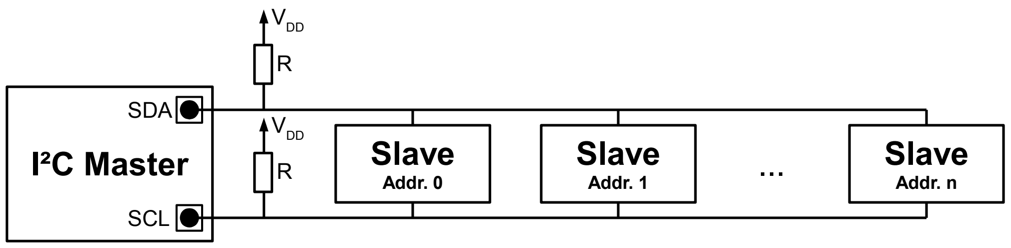


Figure 5.8: I^{2}C Single Master with Multi Slave Connection

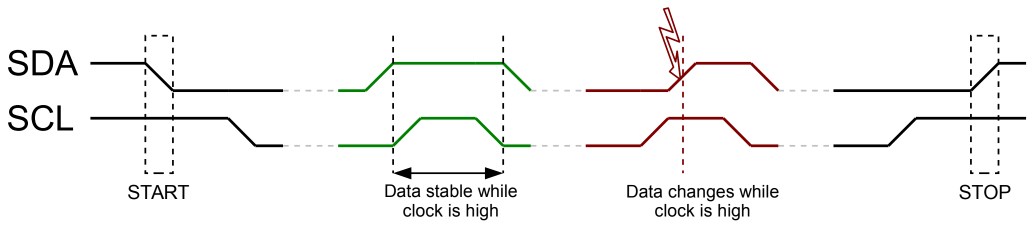

SDA
SCL
START
Data stable while clock is high
Data changes while clock is high
STOP

Figure 5.9: I^{2}C Start/Stop Conditions and (In)Correct Bit Transfer

wire between the master and a slave is connected to V_{DD} over a pull-up resistor. Both sides are able to force the line to signal ground over a transistor, which is used for signal modulation. A logic symbol is represented by the time the line is pulled down and released again (pulse-width modulation). Since the clock line is missing, the synchronization has to happen implicitly. To start a transaction, the master pulls the line down for a certain time (START condition). The low phase of the START condition defines the timing for the bit representation (sampling points). Every bit symbol, independently from the communication direction, begins with a low phase that is triggered by the master. This allows the slave to synchronize on every single bit. Afterwards, either the master or the slave takes control over the communication wire to generate a valid symbol. The symbol ends with a variable high phase that must be long enough to recover the energy of the capacitor that supplies the slave.

5.1.6 Digital UHF Component

The digital UHF component is responsible for the decoding and execution of received EPC Gen 2 commands. The preprocessing of the UHF signal — the removing of the carrier signal and the demodulation of the subcarrier signal— is done in the analog domain. Besides the extraction of the data signal also the supply voltage is extracted from the reader field and passed to the digital domain. After the analog preprocessing, the UHF data signal (*demod*) is sampled and interpreted by the digital UHF component (Figure 5.12). If a valid EPC Gen 2 command is received, the Command FSM is responsible

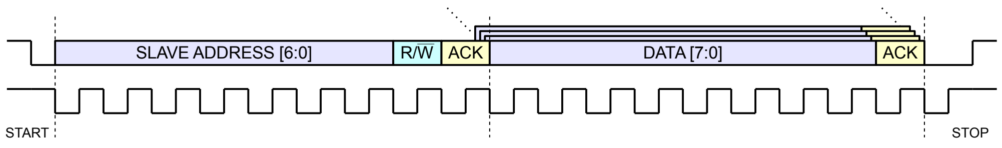


Figure 5.10: I^2C Transaction Format (SDA above SCL signal)

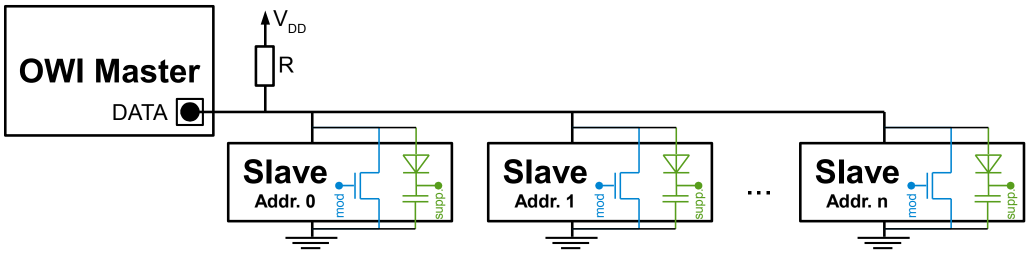


Figure 5.11: OWI Single Master with Multi-Slave Connection

for the extraction and checking of the command parameters and the execution of the received command. The Command FSM is connected to a series of other components, like the Memory Controller (Section 5.1.4), the AES Controller (Section 5.1.2), a CRC generator (cyclic redundancy check), etcetera. Depending on the particular commands and the current state of the EPC Gen 2 FSM, the Command FSM communicates with these components and generates a response. The response is then encoded and passed to the analog part of the chip via the *mod* signal, where this signal is modulated on the carrier signal that comes from the reader field.

5.1.7 Analog UHF Component

Before any data can be processed by the digital part of the EPC Gen 2 tag, the signal has to take a long way through the analog domain. When the tag's dipole antenna enters the reader field, the electromagnetic waves induce a small voltage (Figure 5.13). This voltage is then rectified before a voltage multiplier (charge pump) amplifies the signal. The received energy is saved in a capacitor and powers the voltage regulator that serves as a stabilized voltage source for the other chip components. The band gap works as a reference voltage source for the voltage regulator. When the chip is supplied by the voltage regulator, the Power-On-Reset component initializes the digital components with a generated active-low reset signal. In order to extract the modulated data signal, the subcarrier frequency of the antenna signal is separated from the carrier frequency. This is achieved by an envelope and low-pass filter that is attached to the signal coming out of the voltage multiplier. To finally generate the *demod* signal, which is forwarded to the digital UHF component, the signal amplitude has to be converted to match the digital signal levels. The analog-to-digital signal conversion is performed by a Schmitt Trigger (see Hartl et al. [27] pages 373-378,

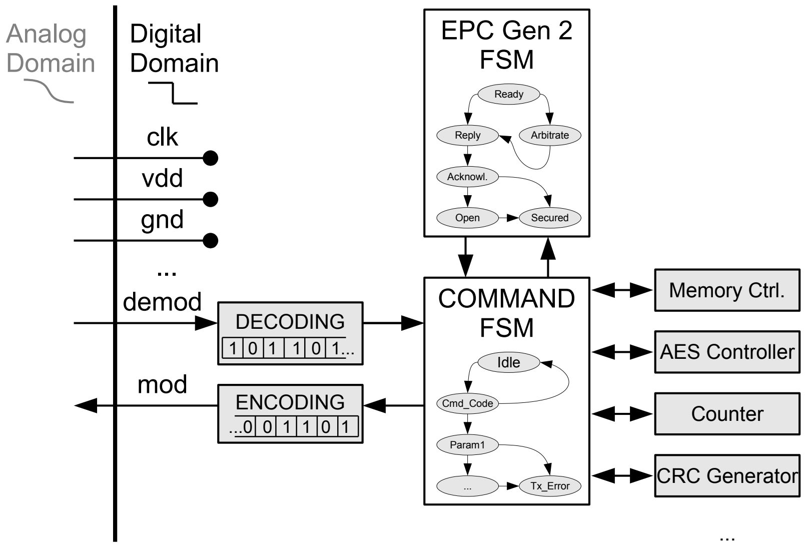


Figure 5.12: Digital UHF Component of the EPC Gen 2 Tag

for further information). This preprocessing step also removes the ripples and produces a clean *demod* signal. The opposite communication way works by influencing the reflected electromagnetic field of the reader (called backscattering). Ideally, the tag absorbs half of the energy that it reflects. By switching a capacitive or resistive load that is connected in parallel to the antenna, the amount of reflected energy is influenced. The change of the reflected energy is recognized and interpreted by a reader. For further information on the communication principles of an electromagnetic-coupled RFID system and a UHF front-end, see Plos[49] pages 12-13 and 18-23. As the EPC Gen 2 standard works with high carrier frequencies —between 860 MHz and 960 MHz— the chip clock is not extracted from the carrier frequency of the reader field. Instead, an integrated oscillator generates the chip clock. Since the clock timing is quite critical, some kind of mechanism needs to be implemented to calibrate the clock. Some tags use so-called phase-locked loops (PLL) to automatically adjust the internal clock timing to match the readers data signal. PLLs are analog modules that regulate a voltage-controlled oscillator source to match the phase and frequency (also multiples) of a source signal. Alternatively, trim bits stored in a non-volatile memory can be used to adjust the timing of the clock generator. Besides the components mentioned and explained above, a variety of other analog components exist that are related to the digital domain but will not be explained any further. These are a random-number generator, other charge pumps and voltage regulators, comparators, etcetera.

5.2 Implementation-Related Problems

Since the starting point of this project was an already implemented and tested EPC Gen 2 tag with an integrated cryptographic module, the very first step was to study the implementation and to find a project entry point. The actual implementation was done in parallel to another big part related to testing and verification, which is discussed on

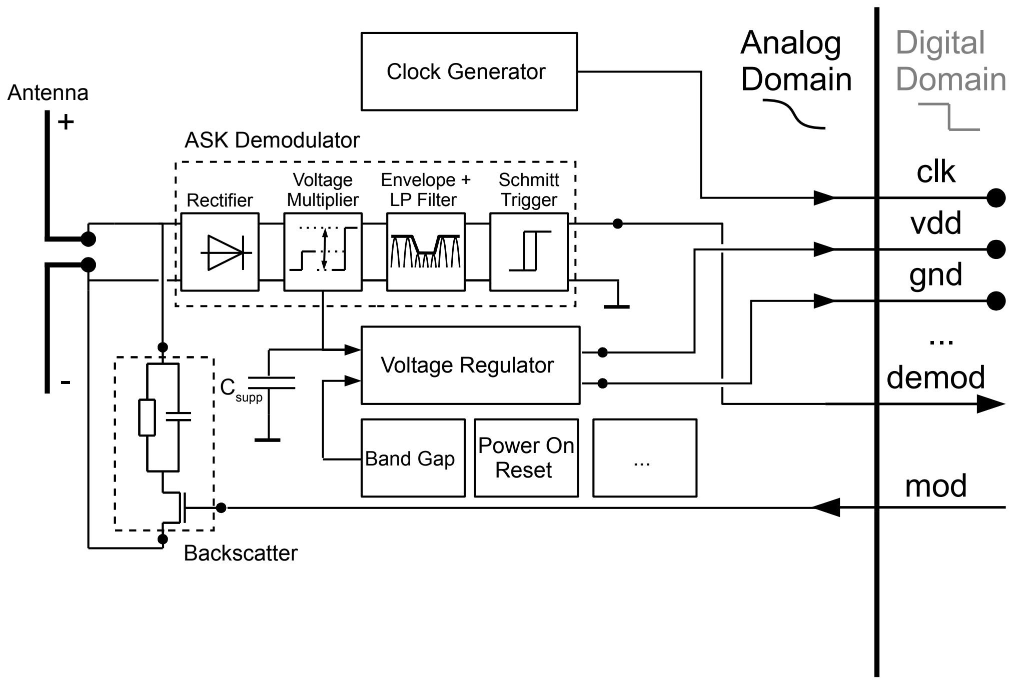


Figure 5.13: Analog UHF Component of the EPC Gen 2 Tag

a theoretical basis in Chapter 4. The last part was the physical implementation of the integrated circuit, which is called back-end design or physical design and is discussed in Section 4.1. Of course a project is never completely straightforward. Concepts that look promising at the beginning of the implementation need to be adjusted sometimes later on to deal with new implemented functionalities. This makes it difficult to subdivide a project into small selfcontained and sequentially processed steps. A retrospective overview of the implementation steps is shown in Figure 5.14 . At the beginning of the implementation, the cryptographic core was removed from the hardware design and replaced by the AES core. Some of the cryptographic-core related parts were not removed completely but adjusted to fit the AES core. The next step was to connect the AES core to the I^{2}C bus. First, the I^{2}C module generated the control and clock signals for the AES core on its own. The AES core interface was mapped into the I^{2}C memory. By reading or writing to this memory region an interrogator was able to access the AES module and its functionality. This approach was good enough to implement the tag authentication over I^{2}C. For the interrogator authentication, another abstraction layer between I^{2}C and the AES core was inserted. The AES Controller implemented not only the finite-state machine that was needed to create a challenge-response authentication protocol (to create a randomized challenge and check the interrogator answer). Also solutions for synchronization of the asynchronous control signals between the I^{2}C module and the AES Controller, and the AES core and the AES Controller were needed. Furthermore, the AES Controller implemented three different clock sources to feed the AES core with a working clock during cryptographic operations. Switching from one clock source to another is not trivial, because it must be ensured that the old clock is not disconnected during its duty cycle and the new clock is not

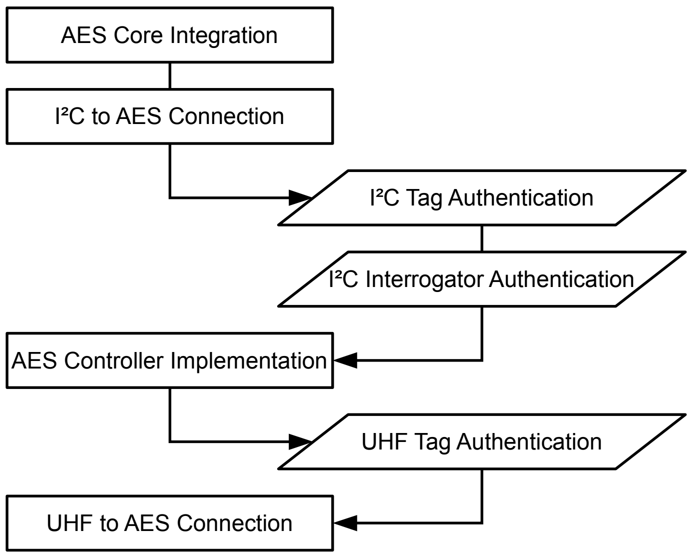


Figure 5.14: Implementation Steps (left) and Implemented Functionalities (right)

faded in during its duty cycle. Both situations would lead to a cropped clock that could cause malfunction of the connected flip-flops. For the UHF tag authentication, the AES Controller was connected to the digital UHF part of the tag. Since the AES Controller works in the same clock domain as the digital UHF part, there was no need to synchronize the control signals. Only the finite-state machine of the AES Controller was extended to serve the implemented UHF functionality. The following section covers problems that occurred during the implementation steps.

5.2.1 Signal Synchronization and Metastability

Since the tag acts as a slave in the I^{2}C communication and the master controls the clock line, it is obvious that the I^{2}C module and the AES Controller work in different clock domains. To put it simple, a chip design where two or more parts work with different clock frequencies is called an asynchronous design. When information exchange is needed between the different clock domains, some kind of signal synchronization needs to be done. Without the synchronization of the signals, setup and hold time violations could occur. Figure 5.15 shows two clock domains, where the first one runs on a slow clock and the second one samples the input *signal* on a faster clock. Transferred to the implementation, the first clock domain represents the I^{2}C module and the second one represents the AES Controller. Since *signal_o* rises and falls at inconvenient times, a direct connection of *signal_o* and *signal_i* would lead to the noted violations. These violations lead —in simple terms— to an unknown sampling result. At first sight, the problem results only in a possibly missed sampling point which is detected at the next rising clock edge. Hence, the synchronization is realized by two sequentially arranged flip-flops that are triggered on the negative edge of *clk2*, which delays the sampling input at least for two clock cycles. So the problem that the synchronization flip-flops combat is not the delay of the signal nor the unknown sampling result. The actual problem is a characteristic of every bistable memory

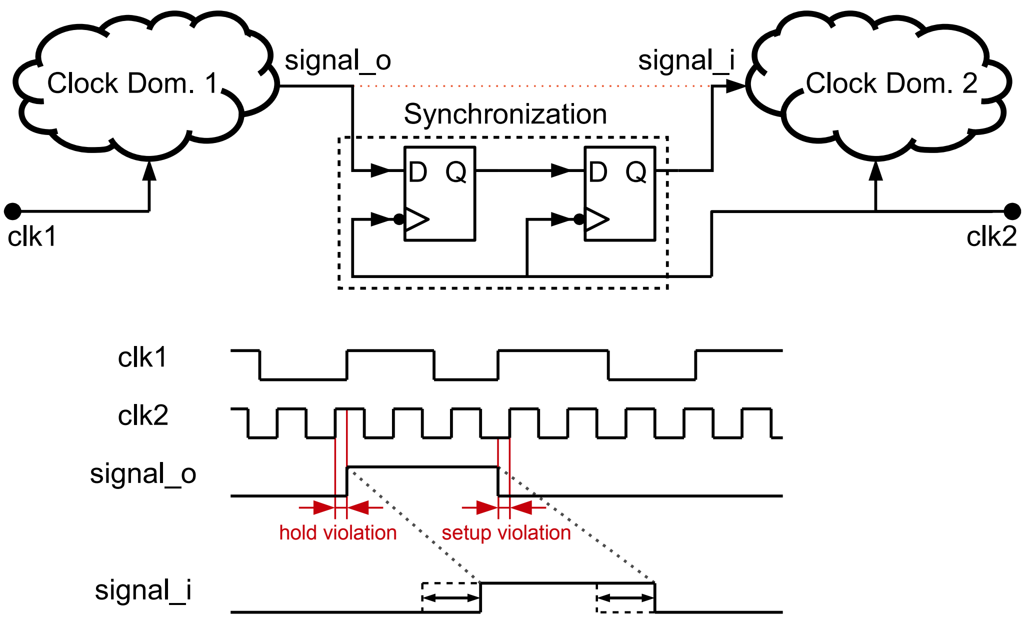


Figure 5.15: Signal Synchronization Between Two Clockdomains

element called metastability and the resulting undefined behavior of the circuit that is connected to it. The problems is quite understandable when the flip-flop is compared to the mechanical system in Figure 5.16. At the beginning the ball lies on the left side of the hill, representing a logic-“1” state of the flip-flop. A certain amount of work is needed to bring the ball uphill over the summit until it finally remains in the “0” position. In a digital integrated circuit there is only a small time window, called setup and hold time, in which the signal strength (cf. force “F”) is applied to the flip-flop state. If the force is not strong enough or long enough applied the ball will not cross the summit. This is the case that can be compared to the missed sampling point. In a worst-case scenario the amount of work is just high enough to bring the ball on top of the hill, where it remains for an unknown amount of time in a state that is neither “1” nor “0”. For the flip-flop this means that the output signal is undefined for a time that could be longer than a whole clock period. Of course the noise, the transistor leakage current and other effects will bring the signal sooner or later to a defined state. But for a connected flip-flop, the metastable signal input could again lead to metastability when the setup time is again violated and so on. The result could be undefined or incorrect behavior of one small part of the integrated circuit that spreads all over the chip. The probability that a flip-flop remains in a metastable state for a time that is long enough to violate the setup time of a connected flip-flop, depends mainly on the clock frequency, the logic path between the flip-flops and the flip-flop properties. A higher clock frequency leads to less time for a signal transition and increases the probability for metastability. Compared to the mechanical system, a higher clock frequency corresponds to a higher and broader hill.
Kaeslin [34] discusses the problem in Chapter 7 (pages 373 - 385) and states clearly that metastability is unavoidable when synchronizers are needed, but the probability that this

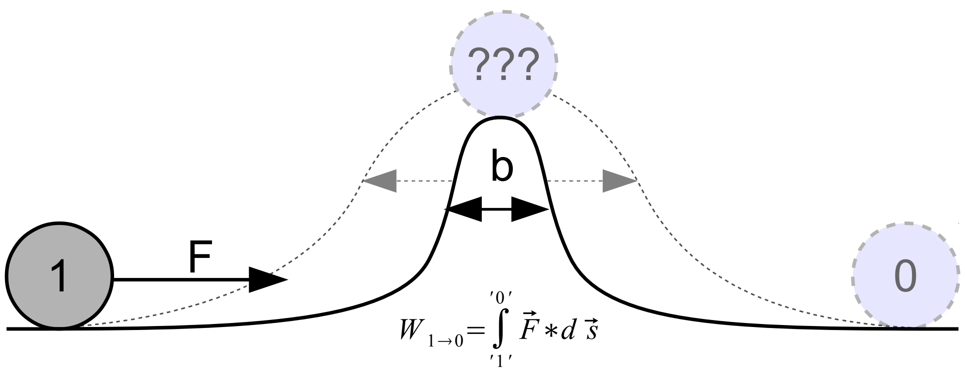


Figure 5.16: Metastability of a Mechanical System

effect occurs can be reduced to a reasonable value. The mean time between errors (MTBE) that are produced by a single-stage synchronizer is calculated by Equation 5.1 (Kaeslin [34] page 380 equation 7.2).

$$t_{MTBE} = \frac{e^{K_2 * t_{al}}}{K_1 * f_{clk} * f_d} \tag{5.1}$$

In this equation f_{clk} is the synchronizer clock frequency, f_d is the frequency of the asynchronous input signal and t_{al} is the allowance time for a state change of the synchronizer flip-flop. The allowance time is calculated as the difference of the synchronizer clock period, the maximum propagation delay (longest combinatorial path) to the next flip-flop and the setup time of this flip-flop (Equation 5.2).

$$t_{al} = T_{clk} - max(t_{prop.delay}) - t_{setupff} \tag{5.2}$$

K_1 and K_2 are the metastability characteristics of the synchronizer flip-flop that should be served by the manufacturers. According to the equation for the MTBE, there are only three factors that designers can use to influence the robustness of the synchronizer. The choice of designated synchronization flip-flops, with known values for K_2 and K_1, is not always available. When the clock frequency can be reduced, this has an enormous effect on the stability of the synchronizer. This can be achieved by an additional clock divider that feeds the synchronizer. The most capable factor is the allowance time for resolving metastability of the synchronizer.
A simple and very effective solution is to use a dual-stage synchronizer that reduces the maximum combinatorial path length behind the first synchronization flip-flop and therefore increases the t_{al} up to a factor of two. Figure 5.17 compares a single-stage synchronizer and a dual-stage synchronizer at different clock speeds. The metastability values for the synchronization flip-flops (K_1 = 100 ps and K_2 = 27.2 GHz) were taken from Kaeslin [34] table 7.2 page 382 for a 130 nm Xillinx XC2VPro4 CLB technology. For the calculation of the MTBE at different f_{clk}, an f_d of 10 MHz, a setup time of the connected flip-flop of 0.5 nanoseconds and a maximum propagation delay of 4 nanoseconds for the single-stage synchronizer were assumed. For the dual-stage synchronizer the assumed propagation de-

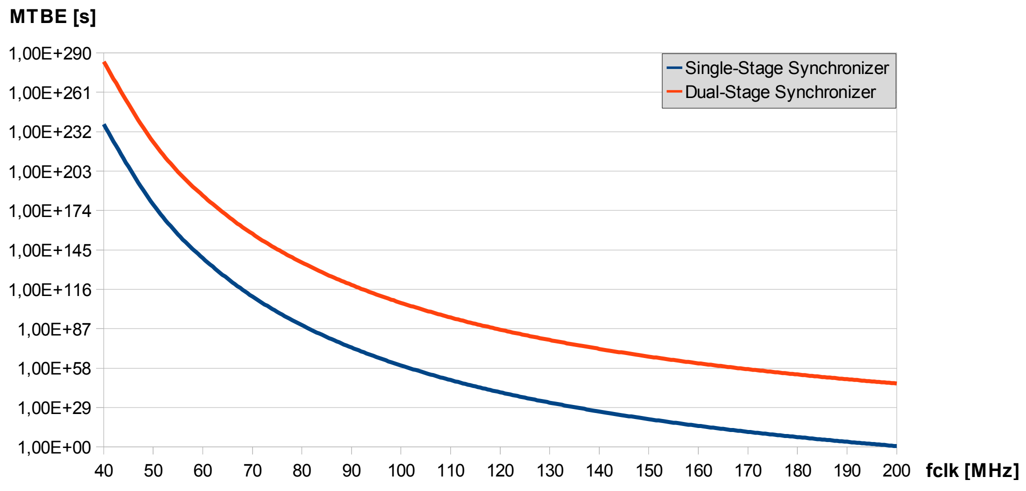


Figure 5.17: "Mean Time Between Errors" for Different Clock Frequencies and for Single-Stage Synchronizers and Dual-Stage Synchronizers

lay was reduced to 0.1 nanoseconds, which causes an increased t_{al}. The graph shows that the f_{clk} has a huge impact on the robustness of the synchronizer. At a clock frequency of 200 MHz the MTBE is just about 4 seconds. By using a dual-stage synchronizer, the MTBE increases to a number that is so overwhelmingly huge that it is very unlikely that this error ever will appear. Besides the dual-stage synchronizers, there are other multi-stage synchronizers that are used when more synchronization robustness is needed, e.g., for higher clock frequencies.

5.2.2 Clock Switching

This section focuses on the problem how to switch between different independent clock signals for a digital component, without generating glitches or cropped clocks. Figure 5.18 shows a naive approach for a clock-switching circuit. Input of the clock-selection multiplexer are three clock signals with different frequency and a signal that selects one of the clocks to be fed forward to the output of the multiplexer. The problems of using this simple solution are obvious. Since a multiplexer consists of different logic gates, any change of the input signals could cause glitches on the output. The other problem is that a change of the *clock_select* signal immediately causes a clock change without taking care of the signal phases. In a worst-case scenario, the clock switching happens while the current clock is at the very beginning of the duty cycle and the selected clock is in a low phase or vice versa (see Figure 5.18). To solve the clock-switching problem, it can be split into two less-complex parts. The solution for the first part of the problem is called **clock-gating** and ensures that only whole duty cycles are faded in on the output of the clock-gating cell. For the second part of the problem —the correctly timed enabling of the clock— the dual-stage synchronizer from Section 5.2.1 is used. The remaining problem is then rather trivial and consists mainly of a logical OR gate that concatenates the outputs of the clock-gating cell (Section 5.2.2).

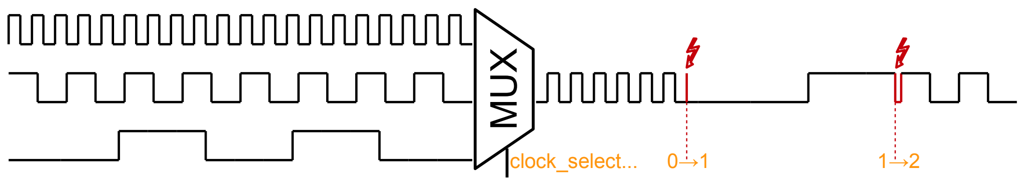


Figure 5.18: Naive Clock-Switching Scheme

Clock Gating

Clock gating is a terminus that is strongly related to power-aware computing, where it is used to dynamically disconnect components from their clock to save power. The power saving is achieved by reducing the switching activity of components like gates or flip-flops. When flip-flops toggle their state, power is consumed through the relatively high charging and discharging currents. By using clock gating, the number of toggling flip-flops per clock edge can be reduced to save power. Even though a flip-flop is disconnected from the clock, a small amount of energy is consumed through a small leakage current between the PN structures. Clock gating is supported by many EDA tools inherently, where optimized clock-gating cells are inserted automatically to save energy. A clock-gating cell has a clock and an enable signal input. The output of a disabled clock-gating cell is logic "0". For an enabled clock-gating cell the output is the correctly gated input clock. Besides the energy saving there are other requirements that a clock-gating cell has to meet. When the clock-gating cell is enabled, the next upcoming clock edge has to be passed to the output immediately. The clock output must be free of glitches and the duty cycle of the input clock must not be cropped by enable-signal glitches. Furthermore, a clock-gating cell needs to be optimized for small delays and power consumption. There are many ways to realize clock-gating. The first approach that would come to ones mind, is the use of logic gates like the approach shown in Figure 5.19, left). It is obvious that this approach is totally unsafe because enable-signal glitches are not filtered and also the state of the clock signal is not taken into account. There are other clock-gating approaches that use only logic gates like multiplexers or NOT/NAND gate combinations but all approaches lack either of robustness or the energy consumption (see Kathuria et al. [36] for a good overview). The use of latches (Figure 5.19, right) for clock gating provides the desired robustness in terms of a clean clock output and has a good energy saving. Hence, the size of a clock-gating cell is about 4.5 GE (according to Kaeslin, page 358). The gate clock signal should be the clock source for multiple flip-flops. Figure 5.20 shows the timing diagram for the clock-gating cells from Figure 5.19. In contrast to the *and_gated_clk* signal that cuts parts of the input clock away, the *latched_gated_clk* contains only complete clock cycles. Even though it is quite easy to build a latch-based clock-gating cell in Verilog or VHDL, it has to be ensured that the inherited clock-gating cells of the according cell library are used. The use of EDA supported clock-gating cells is necessary to guarantee a regular clock distribution with a compensated jitter behaviour.

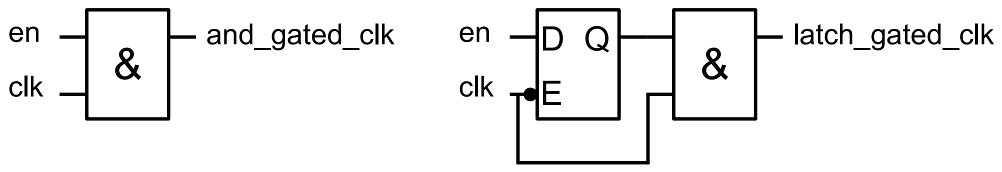


Figure 5.19: Clock Gating with a Logical AND (left) and Latch Based (right)

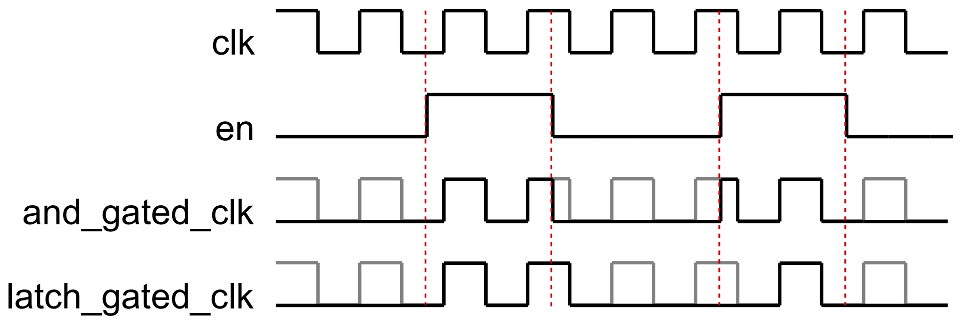


Figure 5.20: Timing Diagram according to Figure 5.19

Clock-Switching Solution

The first step of the clock-switching solution (Figure 5.21) is to synchronize the enable signals coming from a superior controlling instance that runs in a different clock domain than the clock signals on the clock-switching input (e.g., the AES Controller selects the clock source of the AES core by setting one *en* signal). It has to be taken care that only one *en* signal is activated simultaneously. By switching from one clock to another, the Controller is responsible that enough time has passed between switching off the current clock source and activating another one. If there is not enough time in between, two or more clocks could run concurrently and produce an invalid clock signal on the output of the OR gate. After the enable signal has been synchronized to the current clock domain, it is forwarded to a clock-gating cell that takes care, that the chosen clock is neither cropped nor glitches will appear on the output of the clock-gating cell (Section 5.2.2). The clean clock signal is then passed to an OR gate that connects all clock-gating cell output signals. Since only one cell is active, the OR gate delivers a clock signal that is qualified to be the clock input of a component.

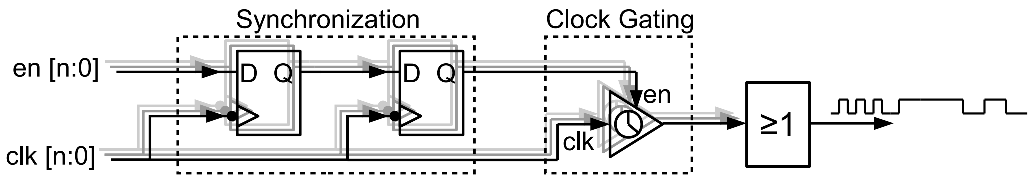


Figure 5.21: Clock-Switching Solution for Multiple Independent Clock Sources

5.2.3 Handshaking

In an asynchronous design, where two modules need to synchronize their work but the clock frequency of one module is not known at the beginning or could also vary, a universal handshaking solution is worthwhile to be found. The idea is to find a solution that is independent from the clock frequencies of the participating modules and can be easily reused for other modules. Figure 5.22 shows a problem that will occur sooner or later when the handshaking between two modules is not implemented properly. It is assumed that the signal-synchronization problem between the two modules (Section 5.2.1) has already been solved. Module B waits for a request to start working and therefore samples the request input signal. Eventually, Module A sends a request and since the time between two sampling points of Module B is less than the request length, the request is detected. While the request is handled, Module A waits for an acknowledge signal to take up its work again. Depending on the time the request was detected, the acknowledge signal will appear inside a certain time window. If the sampling point of Module A does not lay inside this window, the acknowledge is missed and Module A is starving. On the other hand, when the working speed of both modules drifts apart even more, the request signal might appear longer than the handling of the request takes. This could not only lead to a missed acknowledge but also to a request that is handled more than once. The problems also occur when the working frequencies of the modules are exchanged. Therefore, it is necessary to find a way to be independent from the sampling speed in both directions. The approach in Figure 5.23 uses a shared RS flip-flop to buffer the request and acknowledge signals. This ensures that no request or acknowledge signal is lost because of a too low sampling rate. It has to be taken into account that the *req_mst* and *ack_slv* signals must not be high at the same time. To guarantee this, the time for request handling must exceed the request time and the time interval between two requests must be long enough. If this cannot be guaranteed, a further “mutual-exclusion logic” needs to be added (e.g. $[\neg \mathit{req\_mst} \wedge Q]$ between Q and the synchronizer to ensure that *req_mst* must transit to low before the slave gets the signal to start working). The timing diagram in Figure 5.23 shows the signal transitions according to the successful handshake procedure of Figure 5.22. By setting *req_mst* to high the request phase starts and the RS flip-flop saves the request. After two clock cycles the synchronization of the request signal has finished and the slave starts working. During the request-handling phase the next clock edge of the master appears and releases the *req_mst* signal. At the end of the request-handling phase, the slave pulls the *ack_slv* signal to high which triggers the reset of the RS flip-flop. The acknowledge phase lasts from resetting the RS flip-flop until the synchronization of the

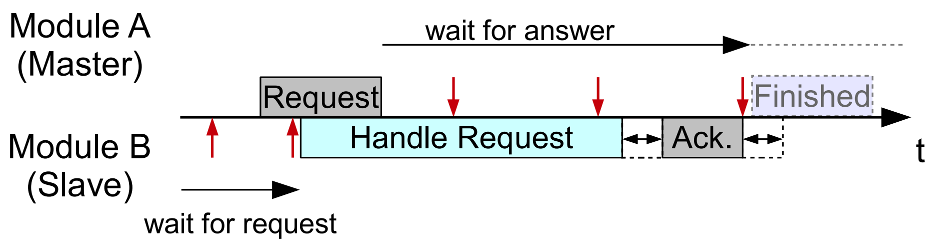


Figure 5.22: Handshaking Problem

negated Q signal is done. This approach can also be used when the clock frequency of the master is higher than the slave's clock frequency. However, it has to be ensured that the *ack_slv* pulse is short enough so it cannot coincide with the next *req_mst* pulse. Otherwise, the mutual-exclusion logic needs to be implemented to assure that the synchronization procedure of the master is delayed until the slave releases the *ack_slv* signal.

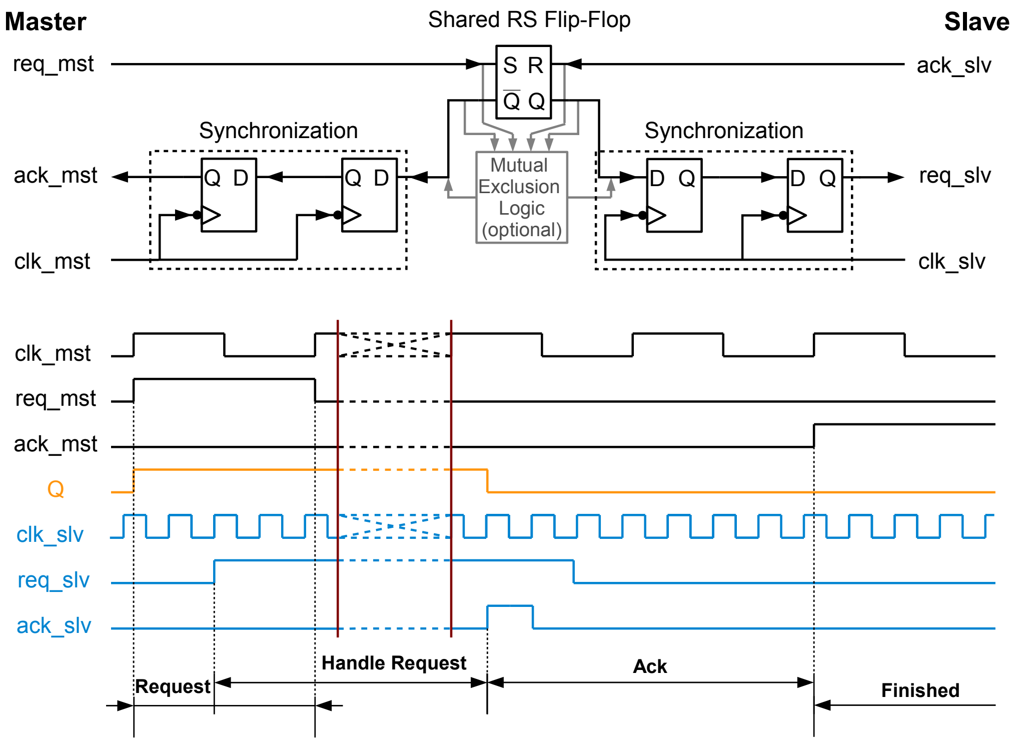


Figure 5.23: Handshaking Solution

Chapter 6

Conclusions

In this thesis, the digital design of a security-enhanced UHF RFID tag was shown. The basis of this work was a digital EPC Gen 2 tag design, which was extended to serve different kinds of secure authentication methods. Authentication methods are used to identify the tag itself or the reader to the other communication party. A secure and effective way for authentication are challenge-response protocols that use symmetric-key cryptography. For these cryptographic operations an AES (Advanced Encryption Standard) core was added to the existing system and connected to a controller. This AES-controller component implements the authentication functionality and the controlling of the AES core by generating the bus signals. Besides the authentication via the air interface, the AES controller also implements the "Tag Authentication" and the "Interrogator Authentication" methods over the I^2C interface. The biggest challenges of the practical part of this work were related to asynchronous design problems. Such problems occur when two or more modules of the digital system are fed with different clocks. At the interface of these modules it has to be ensured that the timing of the input signals is not violated to avoid metastability of the flip-flops which could cause malfunction and undefined behavior of the circuit. Different kinds of asynchronous design problems were explained and solutions discussed.
Classical verification approaches are unfeasible for complex digital designs because the effort for testing every design feature is too high. With classical methods like Tcl scripts or Verilog test benches the estimation of the verification progress gets easier. In this work the design methodologies OVM/UVM and their advantages were elucidated which are the standardization of the verification process, the flexibility and reusability for similar designs, checking of specified but missing functionality, better coverage metrics, etcetera.
In order to provide a broader view on the RFID topic, the thesis was introduced by a historical overview followed by a chapter on the state-of-the-art RFID technology where different fields of applications, different kinds of transponders and relating differentiation features were shown. Since the implemented tag was a UHF tag, this technology was explained in more detail from the generation of electromagnetic waves and the usage as a communication medium to the used communication protocol. The EPC Gen 2 protocol is very complex and supports optional cryptographic commands based on so-called cryptographic suites. To make the implemented UHF authentication functionality clearer an insight was given into the different protocol phases and the related commands. In another part of the work the different ways of implementing authenticity-provisioning methods were discussed and it was stated that challenge-response protocols with symmetric-key cryptography provide a good trade-off between security and implementation costs for a

constrained RFID tag. The AES algorithm was introduced as an example of a symmetric-key block cipher and afterwards it was shown how the cipher can be used to implement different authentication methods. Another part was about privacy and it was stated that this is one of the major problems for the adoption of RFID systems, and because of the missing privacy features the technology still lacks consumer acceptance. It was also stated that security and privacy are not different words with the same meaning and that a secured communication does not automatically preserve privacy. Afterwards, some works on privacy were discussed and physical and protocol-related solutions for privacy preserving were shown. Finally, a hash-based authentication protocol with privacy preserving features was illustrated and described in more detail.
The research on privacy features for RFID tags still seems to be unfinished and a global privacy-standard for RFID applications is missing. In order to become the technology that it was hyped to be and to reach the "plateau of productivity" (cf. Gartner's hype cycle) the privacy issues need to be fixed.

Appendix A

Definitions

A.1 Abbreviations

AES	Advanced Encryption Standard
ASK	Amplitude-Shift Keying
CRC	Cyclic Redundancy Check
DBP	Differential Biphase (Coding)
DES	Data Encryption Standard
DUT	Device Under Test
EAN	European Article Number
EAS	Electronic Article Surveillance
ECC	Elliptic-Curve Cryptography
EEPROM	Electrically Erasable Programmable Read-Only Memory
EIRP	Effective Isotropic Radiated Power
EPC	Electronic Product Code
EPC Gen 2	Electronic Product Code Generation 2
FSK	Frequency-Shift Keying
FSM	Finite-State Machine
GS1	Global Standards One
HDL	Hardware Description Language
HF	High Frequency
HVL	Hardware Verification Language
IEEE	Institute of Electrical and Electronics Engineers
ISBN	International Standard Book Number
ISM	Industrial Scientific and Medical
LF	Low Frequency
Nonce	Number Used Once
NRZ	Non Return to Zero (Coding)
OOP	Object-Oriented Programming
OVM	Open Verification Methodology
PSK	Phase-Shift Keying
RF	Radio Frequency
RZ	Return to Zero (Coding)
RFID	Radio-Frequency Identification
ROM	Read-Only Memory
RSA	Rivest, Shamir and Adleman (Cryptography)

RTL	Register-Transfer Level
SRD	Short-Range Device Frequencies
TID	Tag Identifier
TLM	Transaction-Level Modeling
UHF	Ultra-High Frequency
UID	Unique Identifier
UVM	Universal Verification Methodology
VLSI	Very Large Scale Integration

A.2 Used Symbols

A_E	Effective Aperture
A_S	(Back)Scatter Aperture
c	Speed of Light
f	Frequency
G_i	Antenna Gain Factor
GND	Common Ground Identifier
λ	Wave Length
R_R	Nonce generated by the Reader
R_T	Nonce generated by the Tag
S	Power Density
σ	Radar-Cross Section
T	Signal Period
V_{DD}	Supply-Voltage Identifier

Bibliography

[1] Accellera System Initiative. Standard Universal Verification Methodology Class Reference Manual Release 1.1. Referenced 2013 at `http://www.accellera.org`, 2011.

[2] All About Circuits. Principles of radio : Basic AC Theory. Referenced 2013 at `http://www.allaboutcircuits.com/vol_2/chpt_1/6.html`.

[3] D. Andrews. The irreducible photon : The Nature of Light: What are Photons? III, 742109. In *Proceedings of SPIE*, September 2009.

[4] C. Bell and A. Newell. *Computer Structures: Readings and Examples*. McGraw-Hill computer science series. McGraw-Hill, 1971.

[5] J. Bergeron, E. Cerny, A. Hunter, and A. Nightingale. *Verification Methodology Manual for SystemVerilog*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 0-387-25538-9.

[6] R. Berhorst. Der Urknall : ... und wie die Welt entstand. *GEO kompakt : Die Grundlagen des Wissens*, (29):155, 2011.

[7] A. Bogdanov. Linear Slide Attacks on the KeeLoq Block Cipher. In D. Pei, M. Yung, D. Lin, and C. Wu, editors, *Information Security and Cryptology*, pages 66–80, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-79498-1.

[8] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, September 2007. ISBN 978-3-540-74734-5.

[9] M. W. Cardullo and W. L. Parks III. Transponder Apparatus and System. (Patent 3713148), January 1973.

[10] J. Daemen and V. Rijmen. The Block Cipher Rijndael. In J.-J. Quisquater and B. Schneier, editors, *Smart Card Research and Applications*, volume 1820 of *Lecture Notes in Computer Science*, pages 277–284. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67923-3.

[11] A. Dent and C. Mitchell. *User's Guide to Cryptography and Standards*. Artech House computer security series. Artech House, 2005. ISBN 978-1-58-053530-4.

[12] S. Dominikus, M. E. Oswald, and M. Feldhofer. Practical Security for RFID: Strong Authentication Protocols. In P. D. P. Horster, editor, *D.A.CH Mobility 2006*, pages 187 – 200. Syssec, 2006.

[13] EPCglobal. Class-1 Generation-2 UHF RFID Protocol for Communication at 860 MHZ - 960 MHz. Version 1.1.0. 2005. Referenced 2013 at `http://www.gs1.org/`.

[14] European Parliament. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. Referenced 2013 at `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:NOT`.

[15] K. Finkenzeller. *RFID-Handbuch Grundlagen und praktische Anwendungen induktiver Funkanlagen, Transponder und kontaktloser Chipkarten.* Hanser, 2006. ISBN 978-3-44-640398-7.

[16] K. Finkenzeller and D. Müller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication.* Carl Hanser Verlag, 4nd edition, 2010. ISBN 978-0-470-69506-7.

[17] E. Fleisch and F. Mattern. *Das Internet der Dinge: Ubiquitous Computing und RFID in der Praxis:Visionen, Technologien, Anwendungen, Handlungsanleitungen.* Springer-Verlag New York, Inc., 2005. ISBN 3540240039.

[18] Florian Michahelles from ETH Zürich. When will RFID embrace our everyday lifes?, Invited Talk for the RFIDSec 2012 in Nijmegen. Referenced 2013 at `http://blog.xot.nl/2012/07/03/rfidsec-2012-day-1-summary-of-presentations/`.

[19] Gartner, Inc. Hype Cycle Research Methodology. Referenced 2013 at `http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp`.

[20] P. Gawlik. "Microcomputertechnik" lecture notes. 2002. Referenced 2013 at `http://www.hs-augsburg.de/~bayer/Vorlesungen/mct_download/3SpeicherSS2002.pdf`.

[21] N. Good, D. Molnar, J. M. Urban, D. Mulligan, E. Miles, L. Quilter, and D. Wagner. Radio Frequency Id and Privacy with Information Goods. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, WPES '04, pages 41–42, New York, NY, USA, 2004. ACM. ISBN 1-58113-968-3.

[22] H. Groß and T. Plos. On Using Instruction-Set Extensions for Minimizing the Hardware-Implementation Costs of Symmetric-Key Algorithms on a Low-Resource Microcontroller. In J.-H. Hoepman and I. Verbauwhede, editors, *8th Workshop on RFID Security and Privacy - RFIDsec 2012, Nijmegen, The Netherlands, July 1-3, 2012, Proceedings*, volume 7739 of *Lecture Notes in Computer Science*, pages 149 – 164. Springer-Verlag, 2012.

[23] GS1 - Global Standardization One. GS1 EPC Tag Data Standard 1.6. Referenced 2013 at `http://www.gs1.org/gsmp/kc/epcglobal/tds`.

[24] GS1 - Global Standards One. EPC Gen 2 Standard. Referenced 2013 at `http://www.gs1.org/`.

[25] J. Ha, S. Moon, J. Zhou, and J. Ha. A new formal proof model for RFID location privacy. In S. Jajodia and J. Lopez, editors, *Computer Security - ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 267–281. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88312-8.

[26] D. B. Harris. Radio Transmission Systems with Modulatable Passive Responder. (Patent 2927321), March 1960.

[27] H. Hartl, E. Krasser, G. Winkler, W. Pribyl, and P. Söser. *Elektronische Schaltungstechnik mit Pspice: Mit Beispielen in PSpice.* Pearson Studium. Pearson Studium, 2008. ISBN 978-3-82-737321-2.

[28] IEEE - Institute of Electrical and Electronics Engineers. IEEE Xplore Digital Library. Referenced 2013 at `http://ieeexplore.ieee.org/`.

[29] S. Inoue and H. Yasuura. RFID Privacy using User-Controllable Uniqueness. In *Proc. of RFID Privacy Workshop. MIT*, 2003.

[30] ISO - International Organization for Standardization. Referenced 2013 at `http://www.iso.org/`.

[31] A. Juels. RFID Security and Privacy: A Research Survey. *Journal of Selected Areas In Communication (J-SAC)*, 24(2):381–395, 2006.

[32] A. Juels and R. Pappu. Squealing Euros: Privacy Protection in RFID-Enabled Banknotes. In *Financial Cryptography 03*, pages 103–121. Springer-Verlag, 2002.

[33] A. Juels, R. L. Rivest, and M. Szydlo. The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 103–111, New York, NY, USA, 2003. ACM. ISBN 1-58113-738-9.

[34] H. Kaeslin. *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication.* Cambridge University Press, New York, NY, USA, 1st edition, 2008. ISBN 978-0-52-188267-5.

[35] A. Kamm and M. Baird. *John Logie Baird: A Life.* National Museum of Scotland Publ., 2002. ISBN 978-1901663761.

[36] J. Kathuria, M. Ayoubkhan, and A. Noor. A Review of Clock Gating Techniques. *MIT International Journal of Electronics and Communication Engineering*, 1(2):106–114, 2011.

[37] J. Kilby. Invention of the Integrated Circuit. *IEEE Transactions on Electron Devices*, 23(7):648 – 654, jul 1976. ISSN 0018-9383.

[38] A. Koelle, S. Depp, and R. Freyman. Short-Range Radio-Telemetry for Electronic Identification, using Modulated RF Backscatter. *Proceedings of the IEEE*, 63(8):1260 – 1261, aug. 1975. ISSN 0018-9219.

[39] J. Kuo, J. Lou, and J. Luo. *Low-voltage CMOS VLSI Circuits.* Wiley-Interscience publication. John Wiley, 1999. ISBN 978-0-47-132105-7.

[40] J. Landt. The History of RFID. *Potentials, IEEE*, 24(4):8 – 11, Oct.-Nov. 2005. ISSN 0278-6648.

[41] M. Lenzlinger and E. H. Snow. Fowler-Nordheim Tunneling into Thermally Grown SiO2. *J. Appl. Phys.*, 40:278–283, 1969.

[42] W. Matt, R. van Kranenbury, and G. Backhouse. RFID: Frequency, Standards, Adoption and Innovation. *JISC Technology and Standards Watch,*, May 2006.

[43] G. M. Miller and 1941. *Modern Electronic Communication.* Pearson/Prentice Hall,, Upper Saddle River, N.J. ;, 9th edition, c2008. [Reprinted.]. Includes index.

[44] G. E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8), April 1965.

[45] NIST - National Institue of Standards and Technology. CODATA Value: Speed of Light in Vacuum . Referenced 2013 at `http://physics.nist.gov/cgi-bin/cuu/Value?c`.

[46] OECD - Organisation for Economic Co-operation and Development. OECD Glossary of Statistical Terms - Privacy Definition. Referenced 2013 at `http://stats.oecd.org/glossary/detail.asp?ID=6959`.

[47] R. Pateriya and S. Sharma. The Evolution of RFID Security and Privacy: A Research Survey. In *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, pages 115 –119, june 2011.

[48] Philips Semiconductors (now NXP Semiconductors). I2C-Bus Specification and User Manual - UM10204. 2012. Referenced 2013 at `http://www.nxp.com/documents/user_manual/UM10204.pdf`.

[49] T. Plos. Implementation of a Security-Enhanced Semi-Passive UHF RFID Tag. 2007.

[50] T. Plos, H. Groß, and M. Feldhofer. Implementation of Symmetric Algorithms on a Synthesizable 8-Bit Microcontroller Targeting Passive RFID Tags. In A. Biryukov, G. Gong, and D. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 114–129. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-19573-0.

[51] A. Saxena. *Invention Of Integrated Circuits: Untold Important Facts.* International Series on Advances in Solid State Electronics and Technology Series. World Scientific, 2009. ISBN 978-9-81-281445-6.

[52] P. Söser. ”Integrated Circuits” lecture notes. 2011.

[53] C. Spear. *SystemVerilog for Verification, Third Edition: A Guide to Learning the Testbench Language Features.* Springer Publishing Company, Incorporated, 3rd edition, 2012. ISBN 978-1-46-140714-0.

[54] H. Stockman. Communication by Means of Reflected Power. *Proceedings of the IRE*, 36(10):1196 – 1204, oct. 1948. ISSN 0096-8390.

[55] D.-Z. Sun and J.-D. Zhong. A Hash-Based RFID Security Protocol for Strong Privacy Protection. *Consumer Electronics, IEEE Transactions on*, 58(4):1246 –1252, november 2012. ISSN 0098-3063.

[56] S. Tasiran and K. Keutzer. Coverage Metrics for Functional Validation of Hardware Designs. *IEEE Des. Test*, 18(4):36–45, July 2001. ISSN 0740-7475.

[57] J. Vernon, F. Application of the Microwave Homodyne. *Antennas and Propagation, Transactions of the IRE Professional Group on*, 4(1):110 –116, december 1952. ISSN 2168-0639.

[58] R. A. Watson Watt. Improvements in or Relating to Wireless Systems. (GB593017), November 1935.