Digital RFID Frontend for a Sensor Chip

MA707
Master Thesis

Dino Odobasic

Institute of Electronics (IFE) Graz University of Technology
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Bösch

in cooperation with Infineon Technologies Austria AG

Supervisor: Ass.-Prof. Dipl.-Ing. Dr.techn. Peter Söser
External Supervisor: Dipl.-Ing. Günter Hofer (Infineon)
Graz, in May 2012

This thesis was conducted in cooperation with
Infineon Technologies Austria AG
Development Center Graz
Department for Contactless and RF Exploration
Director: Dipl.-Ing. Gerald Holweg

Abstract

Infineon constantly expands its repertoire of newest chips and technology. One of these chips is an enhanced type of a Radio-Frequency IDentification (RFID) Transponder. Besides standard communication peripherals and integrated memory the Transponder has also integrated sensors to enhance its application possibilities. These applications can be, for example, healthcare applications, like in the Mobile AAL-Systems (MAS) project, which will be further described later on in the document. The Transponder is designed to perform under *low power* conditions. This is achieved by using either a passive (Reader is the power source) or a semi-passive approach (battery as additional power source).

This Master Thesis is related to the development of a digital frontend for the mentioned Transponder. It should be devoted to the computing of incoming datastreams and has to make sure that the Transponder is compliant to the used protocol. So far it supports communication via the EPCglobal Class-1 Gen2 Standard by the existing Comprehensive Transponder System (CTS) [3]. Now another standard should be implemented, enabling the chip to communicate via Near Field Communication (NFC). This technology is standardized, supported and pushed forward by the NFC Forum which consists of many companies around the world. The basis for the NFC Type Tag 2 specification is the ISO 14443 standard, which is responsible for selecting a Tag out of many within the radio frequency field. The selected Tag continues further communication via NFC. That is why a digital communication frontend has to be built, which allows a NFC Forum device to control the Transponders actions.

The development of the frontend begins with the evaluation of several concept ideas. The key factors in the considerations are minimum area and power consumption and high performance. These requirements are best met by using a microcontroller which is responsible for the protocol processing. Additionally a digital component works as an interface between the microcontroller and the sensors and memory. Finally the whole system is synthesized on a 130 nm Complementary Metal Oxide Semiconductor (CMOS) fabrication technology.

Keywords: CTS, ISO 14443, NFC, Reader, RFID, Tag

Kurzfassung

Infineon erweitert stetig sein Repertoire an neuesten Chips und Technologien. Bei einem dieser Chips handelt es sich um eine weiterentwickelte Form eines Radio-Frequency IDentification (RFID) Transponders. Dieser hat zusätzlich zur Kommunikationsperipherie und dem integrierten Speicher auch einige Sensoren, um das mögliche Einsatzspektrum des Transponders zu erweitern. Er eignet sich beispielsweise für Healthcare Applikationen, die im Mobile AAL-Systems (MAS) Projekt vorkommen, auf welches später genauer eingegangen wird. Der Transponder wurde für den Einsatz unter *low power* Bedingungen entworfen. Dies wird durch *passive* (Reader wirkt als Energiequelle) und *semi-passive* (zusätzlich auch Batterie als Energiequelle) Ansätze erreicht.

Diese Diplomarbeit beschäftigt sich mit dem digitalen Frontend des Transponders, welches für die Auswertung der eingehenden Daten und den korrekten Protokollablauf verantwortlich ist. Bisher unterstützt der Chip den EPCglobal Class-1 Gen2 Standard aufgrund des bestehenden Comprehensive Transponder Systems (CTS) [3]. Nun soll ein weiteres Protokoll implementiert werden, welches die Kommunikation über Near Field Communication (NFC) ermöglichen soll, das vom NFC Forum unterstützt und vorangetrieben wird. Die Basis ist hierbei der ISO 14443 Standard. Dieser ist für die Auswahl des Transponders zuständig, falls sich mehrere Transponder im RF-Feld befinden. Ist nun ein Tag ausgewählt, soll die weitere Kommunikation über NFC ablaufen. Daher ist nun ein digitales Kommunikations-Frontend aufzubauen, welches den Tag zusätzlich durch ein NFC-fähiges Gerät (Handy mit NFC-Funktion, etc.) ansteuern lässt.

Die Entwicklung des Frontends beginnt mit der Evaluierung verschiedener Konzeptmöglichkeiten. Die Hauptfaktoren bei der Auswahl des zu implementierenden Konzeptes sind minimale Fläche und Leistungsaufnahme und gleichzeitig hohe Performance. Diese Anforderungen werden bei Einsatz eines Mikrocontrollers erfüllt, welcher für die Protokollabarbeitung zuständig ist. Eine zusätzliche digitale Komponente stellt die Schnittstelle zwischen dem Microcontroller und der Peripherie, wie Sensoren und Speicher, dar. Schlussendlich wird das System mittels 130 nm Complementary Metal Oxide Semiconductor (CMOS) Technologie synthetisiert.

Stichwörter: CTS, ISO 14443, NFC, Reader, RFID, Tag

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)

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 entnommenen Stellen als solche kenntlich gemacht habe.

Graz,am ....

(Unterschrift)

Note of Thanks

This thesis was conducted at the end of my study of the master of science in telematics in cooperation with Infineon Technologies. Many thanks to Dipl.-Ing. Gerald Holweg, Dipl.-Ing. Günter Hofer and my collegues. They supported me in any way and helped me a lot to gain better skills and finish the work quickly.

Furthermore I thank my supervisor, Dipl.-Ing. Peter Söser, who was available at any time when needed. At the end I thank my parents, family and friends who supported me the whole 6 years until I had the possibility to stand on my own feet.

Dino Odobasic

Contents

List of Figures

List of Tables

Chapter 1: Introduction

This Master Thesis is part of the MAS Project. The coordination is performed by Infineon Technologies AG. The Department of Contactless and Radio Frequency Exploration (CRE) at the Design Center Graz, Austria is responsible for the development of *active*, *passive* and *semi-passive* prototypes of contactless RF-Systems. Therefore CRE develops a prototype chip for the contactless sensor applications characterized by MAS.

This chapter presents an overview of the MAS Project in Section 1.1. Section 1.2 focuses on the objective of the Thesis, while Section 1.3 describes the structure of the Thesis.

1.1 Motivation - The MAS Project

MAS stands for Mobile AAL Systems and defines the development of a common communication platform and nanoelectronic circuits for health and wellness applications. The objective of AAL (Ambient Assisted Living) is to enhance the quality of human life and improve the wellbeing of people. Five application scenarios are described by MAS:

1. Health and Activity Monitor
2. Point of Care Terminal and Gateway
3. Cardiovascular Monitor
4. Diabetes Monitor
5. Mobile Cardiotocography

Scenario five is shown in Figure 1.1 and describes a possible application for a contactless sensor chip. Sensors should gather different vital signs, like heart rate or temperature over time. The recorded data can be displayed on a lightweight PDA-size device. This device can be an NFC Forum device, like a mobile phone with NFC functionality. The communication with the sensor should be established by placing the device close to the sensor. The transmission of the data for further processing should happen via a WiFi or Mobile Network. As shown in Figure 1.1, the information can be transmitted through the cloud to doctors or paramedics. Therefore they can react immediately.

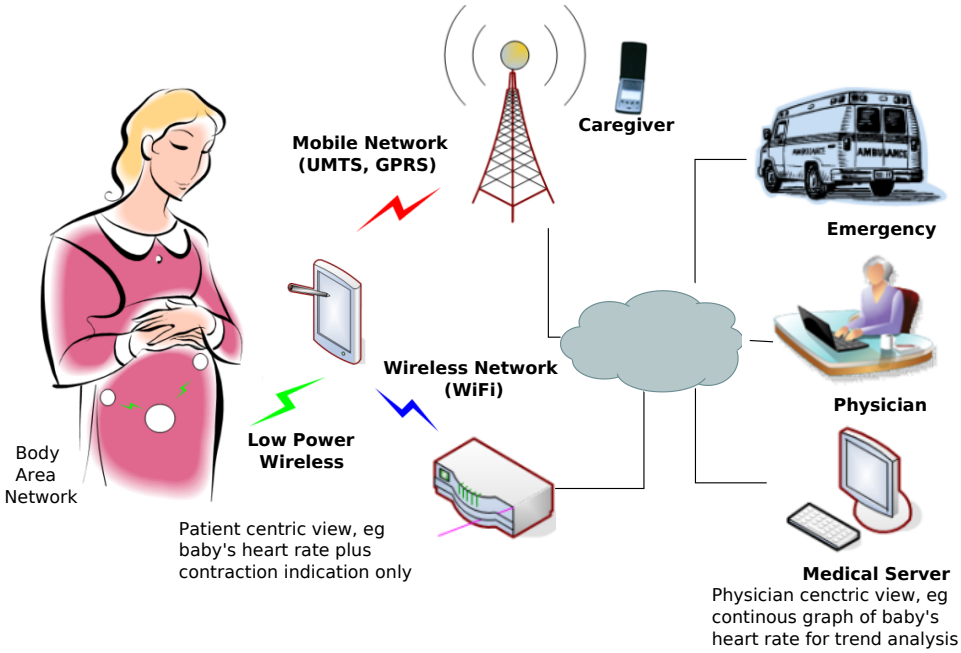

Figure 1.1: Szenario 5

Adopted from [1]

A more precise definition of each scenario and the whole project can be found on the website of MAS [1].

1.2 Aim of the Thesis

This Master Thesis covers some application scenarios of the MAS Project, because the chip can be used in more than one scenario. Figure 1.2 provides a conceptual overview of the chip. It's provided with energy whether from an external source or from a Thermo-harvester, which obtains its energy from temperature changes. The external source can also be an on-chip battery.

Two Analog Front Ends (AFE) enable the Transponder to communicate via EPC or NFC. If a device is placed close to the chip it receives the energy from the device via the electromagnetic field or antenna (*passive* energy supply). Additionally a Thermo-Harvester can be responsible for the energy supply (*semi-passive* solution). After the

chips Protocol Machine or Microcontroller has established the communication sensors can be activated to get data via the sensor interface. The application scenario determines the sensor and the data obtained. Data is stored in a Non Volatile Memory (NVM) for further usage. The security module prevents hostile data manipulation. Finally an external digital interface (e.g. Serial Peripheral Interface (SPI)) enables the chip to communicate with other sensors or devices.

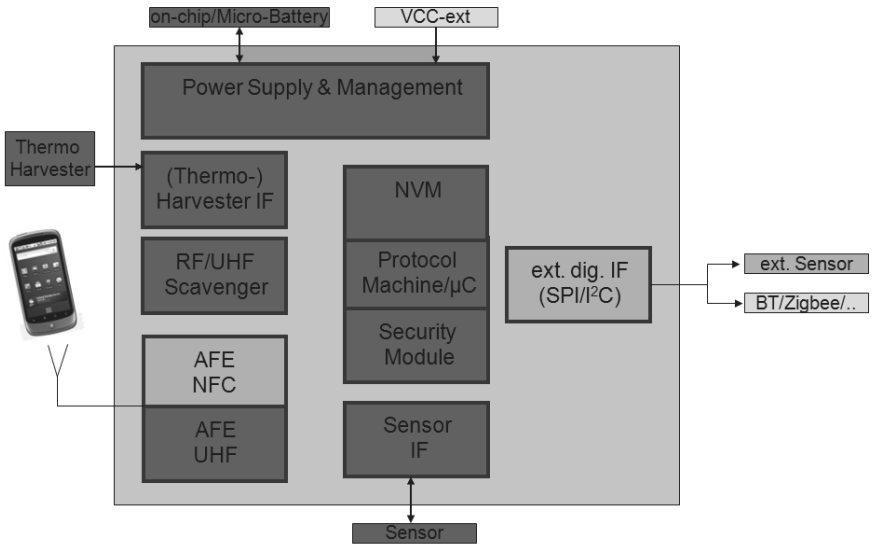

Figure 1.2: Chip Overview

Adopted from [8]

Until now communication is limited to EPCglobal Class-1 Gen2. The disadvantage of the current chip is the necessity of an extra Reader Hardware. By contrast, NFC is implemented in many mobile devices, like cell phones or smartphones. This assures a fast and easy way to read out sensor data.

At first the NFC protocol processing must be implemented in Firmware on a *low power* Reduced Instruction Set Computer (RISC) for RFID applications. The next step is to connect the RISC to the already existing internal bus, so that it can communicate with the NVM, sensors and external devices. Finally the system has to be synthesised using a 130 nm CMOS fabrication technology.

1.3 Thesis outline

Chapter 2 consists of a brief explanation of RFID technology and the used protocols ISO 14443 and NFC Type Tag 2. It describes the initialization and anticollision (selecting one Tag out of many) mechanism of ISO 14443 and the communication protocol used by NFC Forum devices. In order to establish a communication with the Host-System (Reader) these mechanisms have to be implemented in the Microcontroller.

Chapter 3 provides at the beginning an overview about digital design generally and some relevant considerations to ensure a functional system. Later on it focuses on the design of the components, which are used for communication via NFC. It covers also their interconnection and the principle how they are supposed to work together.

In Chapter 4 the results of the Register Transfer Level (RTL) simulations of the designed components are discussed. They should correspond with the expected behaviour as it is described in chapter 2 to ensure safe communication between Reader and Tag.

Chapter 5 illustrates the results of the synthesis of the implemented digital units.

In Chapter 6 measurements are compared with the simulations.

Finally a summary of the results and an outlook is given in Chapter 7.

Chapter 2: Basics

Section 2.1 of this Chapter describes the basic concept of an RFID system. Furthermore it offers an insight into the protocols in sections 2.2 and 2.3, which must be implemented. This includes modulation schemes, bit representation, Tag selection principles and the communication process itself.

2.1 Radio-Frequency IDentification (RFID)

An RFID system is a wireless communication system. It consists of 2 devices called Reader and Tag. Figure 2.1 shows such a system. The Reader is connected with a host, which may be a Personal Computer (PC) or a mobile device, but also the host and Reader can be the same device. Reader and Tag are communicating wirelessly via antennas. The Tag or Transponder is responsible for data aquisition. This data is then delivered to the Reader, if it requests it [4].

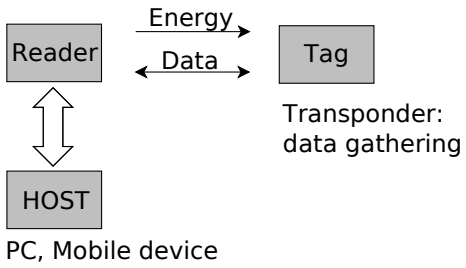

Figure 2.1: RFID System

The communication is based on the inductive coupling principle. The Readers antenna generates an electromagnetic field which transports the data and energy for the Tag. The coding scheme may be a Modified Miller with 100% modulation, where data is represented as a variation of the amplitude of the signal (Amplitude Shift Keying - ASK). If the Tags resonance frequency corresponds with the Readers transmission frequency, the Tag draws additional energy from the field. By switching on and off the load resistance

of the Tag, this power consumption can be measured at the readers antenna. This is called load modulation and can be used to transmit data from Tag to Reader [4].

2.2 ISO 14443

The ISO 14443 standard consists of four parts [2].

- Part 1 describes the physical characteristics of a proximity card (PICC).
- Part 2 specifies the radio frequency and power characteristics of a signal from a Proximity Coupling Device (PCD) to a PICC.
- Part 3 describes the byte format and framing and the anticollision procedure to communicate with multiple PICCs in the field.
- Part 4 describes the transmission protocol.

This thesis uses information and knowledge from parts 2 and 3. Two standards are defined, which use different modulation schemes. The selected scheme uses the Type A interface for communication, which will be described next.

2.3 Communication signal for Type A interface

For a carrier frequency f_c of 13.56 MHz the bit rate during initialization and anticollision should be $f_c/128$ (approx. 106 $kbit/s$). The subcarrier frequency during load modulation should be 848 kHz [13]. The frequencies and timings are delivered by the Contactless Universal Asynchronous Receiver Transmitter (CLUART), which will be introduced later. Figure 2.2 shows the used modulation scheme and signals, which are used for establishing a communication between PCD and PICC.

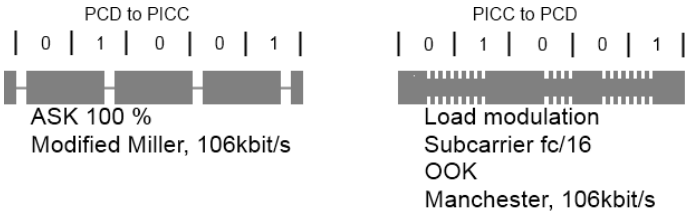

Figure 2.2: Type A: signals and modulation schemes for both directions

Adopted from [13]

Timing constraints are shown in Table 2.1 and the pause, which is created during ASK 100% is illustrated in Figure 2.3. For a correct functioning, the time parameters $t_1 - t_4$ have to be met [13].

	Condition	**Min**	**Max**
t_1		2.0 μs	3.0 μs
t_2	$t_1 > 2.5\ \mu s$	0.5 μs	t_1
	$t_1 \leq 2.5\ \mu s$	0.7 μs	
t_3		0	1.5 μs
t_4		0	0.4 μs

Table 2.1: Timing parameters for a pause

Adopted from [13]

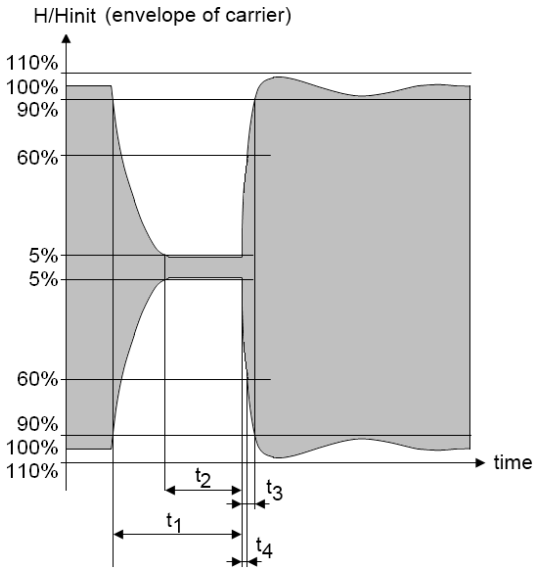

Figure 2.3: Pause Timing

Adopted from [13]

Overshoots should be kept between 90% and 110%. The end of a pause should be detected after the field exceeds 5% and before it exceeds 60% [13].

2.3.1 Bit coding of a signal from PCD to PICC

At ASK 100% the data is represented as variation of the amplitude of the carrier. As shown in Figure 2.4 the bits, Start Of Frame (SOF) and End Of Frame (EOF) are coded with the Modified Miller coding scheme. Bit "1" consists of a pause after a time of half the bit duration. The coding for bit "0" depends on the previous data. If it is transmitted after a "1", no modulation should occur. If there are more than one "0"s, the second and following "0"s are coded with a pause at start of the bit duration. This applies also to the case, that an SOF precedes the "0". An EOF is coded as a "0" and no modulation for one bit duration. No information is signaled with 2 consecutive sequences with no modulation [13].

Figure 2.4: Bit coding from PCD to PICC

2.3.2 Bit coding of a signal from PICC to PCD

The Manchester coded signal from Tag to Reader is also described more precisely now. The coding scheme is shown in Figure 2.5. Bit "1" and SOF are defined by a load modulation of the signal for the first half of a bit duration. Bit "0" is defined by a load modulation of the signal for the second half of a bit duration. No modulation for one bit duration signals EOF. No subcarrier signals no information [13].

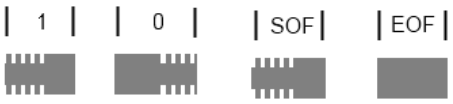

Figure 2.5: Bit coding from PICC to PCD

2.4 Initialization and Anticollision protocol

The PCD starts the communication with a specified start of communication signal, which corresponds to SOF. It transfers the information and signals end of communication with an EOF, as described in the last section. Afterwards a Frame Delay Time (FDT) between the last pause transmitted by the PCD and the first modulation edge transmitted by the PICC has to pass. This delay depends on the command type and the logic state of the last transmitted data bit in this command. Figure 2.6 and Table 2.2 illustrate this correlation [14].

command type	**n (integer value)**	**FDT**	
		last bit = “1”	last bit = “0”
REQA, WUPA ANTICOLLISION SELECT	9	1236 / f_c	1172 / f_c
ALL OTHER	≥ 9	$(n * 128 + 84)/f_c$	$(n * 128 + 20)/f_c$

Table 2.2: Frame Delay Time FDT

Adopted from [14]

The delay time between the last modulation transmitted by the PICC and the first pause transmitted by the PCD should be at least $1172/f_c$. The Request Guard Time of $7000/f_c$ is the minimum time between 2 start bits of 2 successive Request commands.

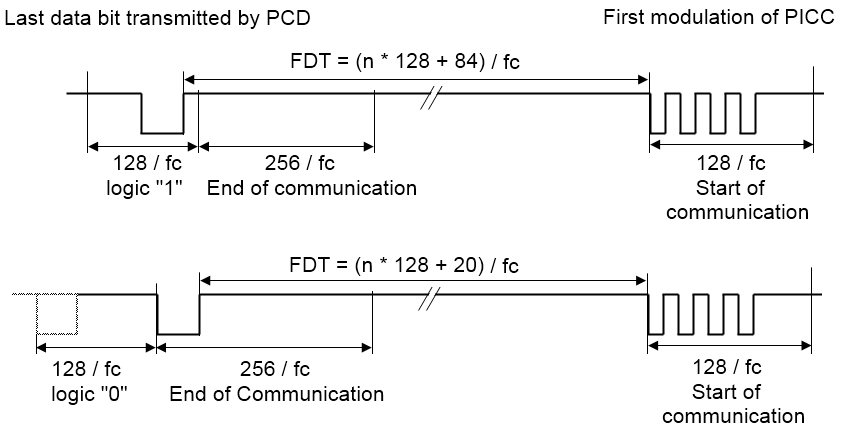

Figure 2.6: Illustrated delay time

Adopted from [14]

The value n in Table 2.2 means that all PICCs in the field should respond in a synchronous way. This is needed for anticollision, where the PCD receives Manchester coded signals. Their logic states are defined in Figure 2.7 and depend on level changes and not on the level itself. This allows the recognition of transmission errors. In this case these errors are caused by 2 or more PICCs in the field, which are responding at the same time [4],[14].

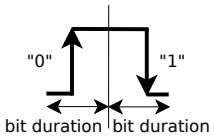

Figure 2.7: Manchester coding

So one bit duration must contain level changes. If not, 2 or more Transponders are operating in the field and a collision occurs, as illustrated in Figure 2.8 [4].

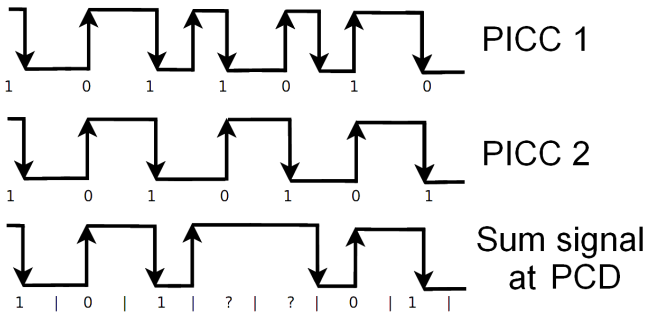

Figure 2.8: Anticollision with Manchester coding

Adopted from [4]

Now the initialization and anticollision algorithm starts. Figure 2.9 displays the state diagram for a PICC during anticollision. A PICC has several states. These are POWER OFF, IDLE, HALT, READY and ACTIVE. After a reset the IDLE state is reached. With an initialization command the PICC changes its state to READY. Now select and anticollision commands ensure the selection of only one PICC, if more than one operate in the field. After the Reader or PCD has selected one PICC, the ACTIVE state is reached. Now the PICC fulfills its duties and changes its state to HALT if the work is done or an error has occured. Certain commands in states lead to errors, so that the PICC returns to either IDLE or HALT state.

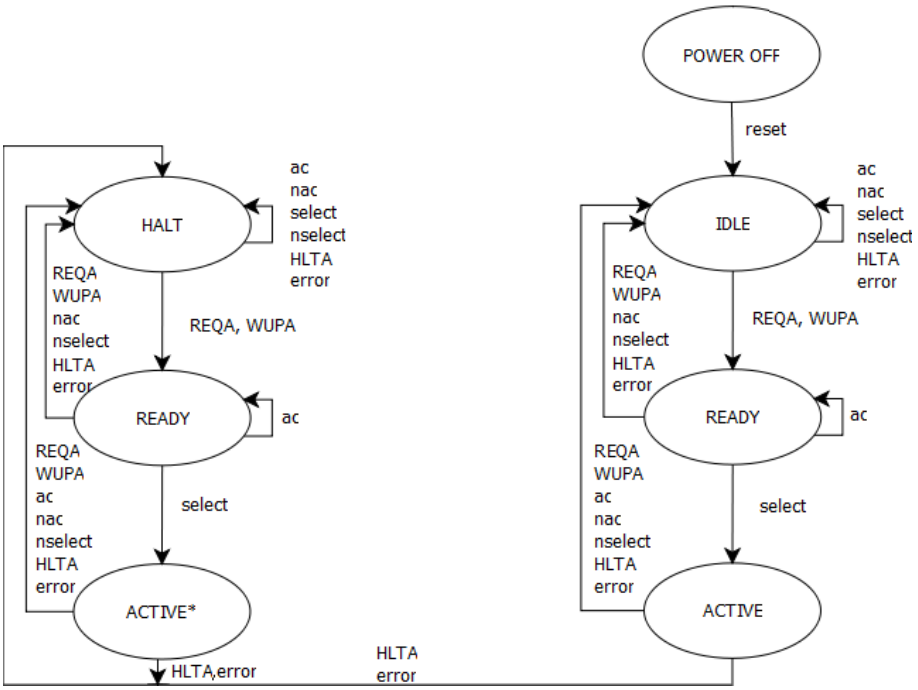

Figure 2.9: PICC states

Adopted from [14]

2.4.1 Commands (PCD to PICC)

This section describes the commands, which are used by the PCD for initialization and anticollision in more detail.

2.4.1.1 REQA and WUPA

After a reset the PICC is powered and listens to REQA (Request Type A) and WUPA (Wake Up Type A) commands, as shown in Figure 2.9. If any other command is transmitted by the PCD, it will be ignored and the PICC is kept in IDLE state. These commands consist of a short frame with the length of 7 bits, as illustrated in Figure 2.10. There is also a start of communication at beginning and end of communication at the end [14].

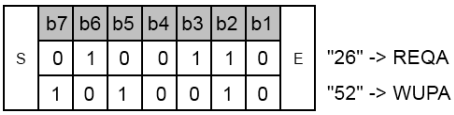

Figure 2.10: REQA and WUPA short frame

Adopted from [14]

2.4.1.2 Select and anticollision commands

The terms ac and nac in Figure 2.9 represent anticollision commands. Select and nselect represent select commands. They are built up using a so-called standard frame. It consists of a start of communication, $n * (8\ data\ bits + odd\ parity\ bit)$ and an end of communication. Its structure is shown in Figure 2.11 [14].

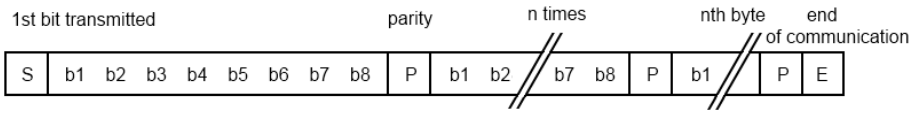

Figure 2.11: Structure of a standard frame

Adopted from [14]

These anticollision commands and select commands consist of:

- 1 byte select code (0x93 for cascade level 1, 0x95 for cascade level 2)
- 1 byte with the Number of Valid Bits (NVB) before a collision
- 0 to 40 data bits which contain the Unique IDentification (UID) of the current cascade level according to the value of NVB.

A third cascade level is available, but is not used here.

An UID may consist of 4 to 10 bytes. Here the UID is made up of 7 bytes. Thus the first byte must be a Cascade Tag, which will be explained later. The first UID byte contains the manufacturer ID, whichs value must not be in the range of 0x81 to 0xFE. These are marked for private use. The remaining bytes are defined by the manufacturer. Figure 2.12 and Figure 2.13 illustrate a collision and the corresponding anticollision algorithm. It uses a binary search technique to select particular PICCs in the field. If a collision occurs at the end of a byte, the PCD transmits an anticollision command and all databits received

before the collision to select specific PICCs. In this example, all PICCs with UID0 = 0x32 and UID1 = 0x10 respond with the remaining UID followed by a checkbyte (BCC). The checkbyte will be explained later. The remaining PICCs are kept in IDLE state.

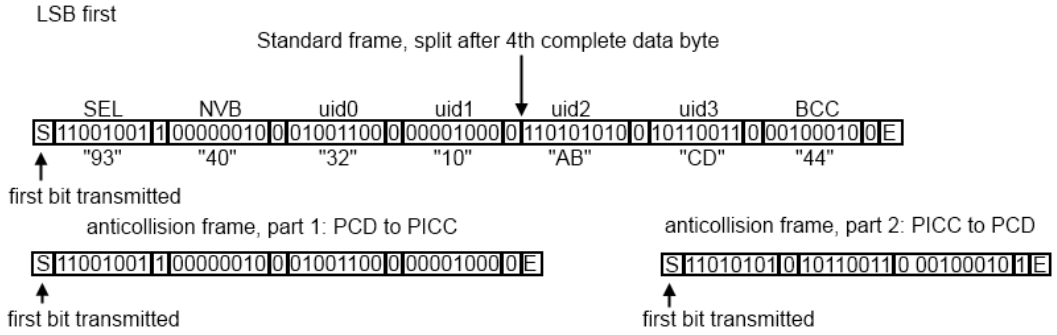

Figure 2.12: Split after full byte

Adopted from [14]

It is the same procedure, if a collision occurs within a byte but now no parity bit is added after the anticollision frame transmitted from PCD to PICC. Also the first parity bit, which is added after the remaining bits from the PICC is not taken into account by the PCD.

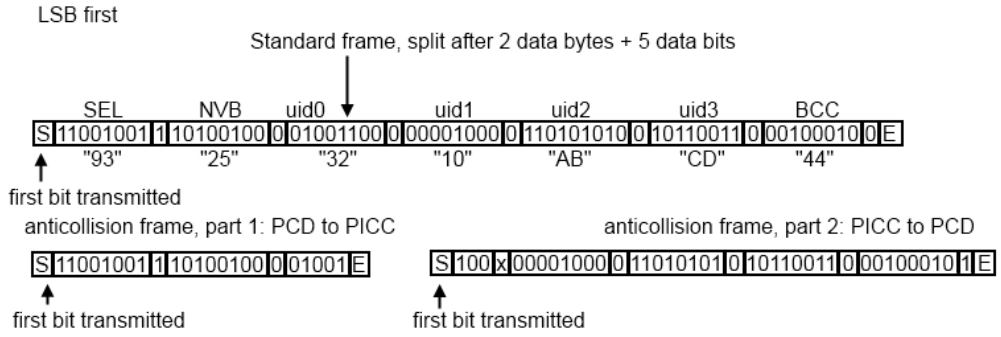

Figure 2.13: Split after 2 bytes and 5 bits

Adopted from [14]

2.4.1.3 HLTA command

This command changes the PICCs state to HALT, which is nearly the same like IDLE, but now the PICC responds only to a WUPA command. Its structure is shown in Figure 2.14 [14]. It consists of 2 data bytes and 2 Cyclic Redundancy Check (CRC) bytes. The CRC will be explained later.

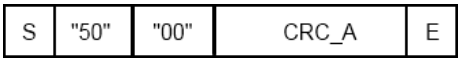

Figure 2.14: HLTA command

Adopted from [14]

2.4.2 Responses (PICC to PCD)

Once the PCD transmits a command, the PICC reacts with a response, which will be described in this section. They depend on the received command, the UID of the PICC and the current state.

2.4.2.1 ATQA

All PICCs in IDLE or HALT state should respond synchronously with ATQA (Answer To Request Type A) after a REQA or WUPA command. Now the PICC changes its state to READY and the PCD transmits a select command. The structure of ATQA is shown in figure 2.15 [14].

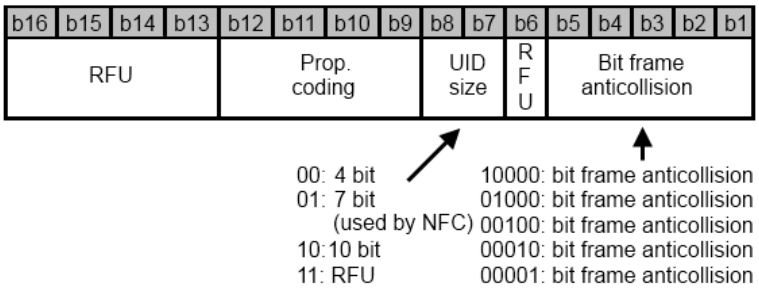

Figure 2.15: ATQA response

Adopted from [14]

The first 5 bits declare that bit frame anticollision is active. The UID size is set to “01” to signal a UID size of 7 bytes. The remaining bits are Reserved for Future Use (RFU), because proprietary coding is not required.

2.4.2.2 SAK

In READY state, the PCD may send three possible commands which provoke responses as illustrated in Figure 2.16.

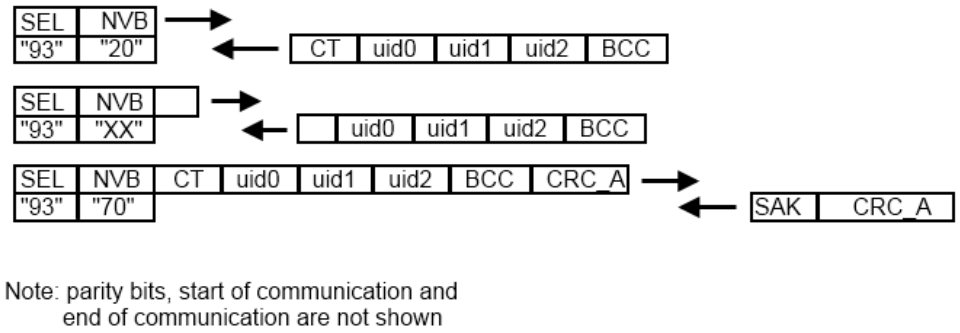

Figure 2.16: Select and anticollision responses

Adopted from [14]

The first anticollision command in Figure 2.16 contains the value 0x20 for NVB, which specifies that no part of the UID is transmitted. The response of the PICC consists of its UID at the specified level (0x93 for cascade level 1, 0x95 for cascade level 2) and its BCC. The BCC is a logical EXCLUSIVE OR operation of all UID bytes at a level. Thus its value only depends on the value of the UID. CT is the cascade tag, which is needed if more than one cascade level is used. This is the case in the NFC Type Tag 2 standard. So the first byte of an UID is 0x88 in order to signal that also another cascade level is used to store the second part of the UID. If there is only one cascade level, the first byte must not contain the value 0x88. As long as collisions occur the PCD transmits anticollision commands to select one PICC. The PCD can select it with transmission of a select command. It consists of a SEL byte and an NVB of 0x70, which specifies that 5 bytes are following. These 5 bytes are made of 4 UID bytes and the BCC. Finally 2 bytes, which contain the CRC are attached. The CRC bytes are not considered in the NVB. The PICC responds with a byte called SAK (Select Acknowledge) followed by 2 CRC bytes. Its content is shown in Figure 2.17 [14].

b8	b7	b6	b5	b4	b3	b2	b1	
x	x	x	x	x	1	x	x	=> cascade bit set: UID completed
x	x	1	x	x	0	x	x	=> UID complete, PICC compliant with 14443-4
x	x	0	x	x	0	x	x	=> UID complete, PICC not compliant with 14443-4

Figure 2.17: Select Acknowledge (SAK)

Adopted from [14]

Here the Tag is not compliant with ISO 14443-4, so the 6^{th} bit is kept "0". After an SAK response the PICC changes its state to ACTIVE. Now it accepts READ, WRITE and HLTA commands, which are part of the NFC Forum Type 2 Tag Technical Specification.

2.5 NFC Type 2 Tag Technical Specification

As explained before, the NFC standard uses ISO 14443 as Tag selection protocol. After a Tag was selected it is in ACTIVE state and responds to NFC commands. The next section will explain the used memory structure and the so-called NFC Data Exchange Format (NDEF). Messages, which are transmitted via NFC are embedded in this format. Afterwards the NFC commands will be explained.

2.5.1 Memory structure

If the memory size is equal to 64 bytes, the Tag uses a static memory structure. In case of the memory size is bigger than 64 bytes, the Tag uses a dynamic memory structure. The static memory structure is described in detail in the NFC Forum Type 2 Tag Technical Specification. For this thesis a dynamic memory structure is used and will be discussed now in detail [5].

The memory is built up from 24 blocks with 4 bytes each, as shown in Figure 2.18. The data area consists of 20 blocks.

	ByteNumber				BNo	Addr
Content	0	1	2	3	dec	hex
UID|BCC0	UID0	UID1	UID2	BCC0	0	55|56
UID_2nd_part	UID3	UID4	UID5	UID6	1	57|58
BCC1|Int|Lock	BCC1	Internal2	Lock0=0x00	Lock1=0x00	2	59|5A
CC	CC0=0xE1	CC1=0x10	CC2=0x0A	CC3=0x00	3	5B|5C
LockControlTLV	**LCTLV0=0x01**	**LCTLV1=0x03**	**LCTLV2=0xC0**	**LCTLV3=0x04**	**4**	**5D|5E**
LockCtrlTLV|MCTLV	**LCTLV4=0x33**	**MCTLV0=0x02**	**MCTLV1=0x03**	**MCTLV2=0xC1**	**5**	**5F|60**
MCTLV|NDEFMsgTL	**MCTLV3=0x03**	**MCTLV4=0x03**	**NDEFT=0x03**	**NDEFL=0x42**	**6**	**61|62**
NDEFV	**NDEFV0=0xD1**	**NDEFV1=0x01**	**NDEFV2=0x3E**	**NDEFV3=0x55**	**7**	**63|64**
NDEFV|NDEFMsg[...	**NDEFV4=0x00**	**NDEFSKIPNULL**	**SENSOR_CONFIG00**	**SENSOR_CONFIG00**	**8**	**65|66**
config	**SENSOR_CONFIG01**	**SENSOR_CONFIG01**	**SENSOR_CONFIG02**	**SENSOR_CONFIG02**	**9**	**67|68**
config	**SENSOR_CONFIG03**	**SENSOR_CONFIG03**	**SENSOR_CONFIG04**	**SENSOR_CONFIG04**	**10**	**69|6A**
config	**SENSOR_CONFIG05**	**SENSOR_CONFIG05**	**SENSOR_CONFIG06**	**SENSOR_CONFIG06**	**11**	**6B|6C**
config	**SENSOR_CONFIG07**	**SENSOR_CONFIG07**	**SENSOR_CONFIG08**	**SENSOR_CONFIG08**	**12**	**6D|6E**
config	**SENSOR_CONFIG09**	**SENSOR_CONFIG09**	**SENSOR_CONFIG10**	**SENSOR_CONFIG10**	**13**	**6F|70**
config	**SENSOR_CONFIG11**	**SENSOR_CONFIG11**	**SENSOR_CONFIG12**	**SENSOR_CONFIG12**	**14**	**71|72**
config	**SENSOR_CONFIG13**	**SENSOR_CONFIG13**	**SENSOR_CONFIG14**	**SENSOR_CONFIG14**	**15**	**73|74**
config	**SENSOR_CONFIG15**	**SENSOR_CONFIG15**	**sensorSTART**	**sensorSTART**	**16**	**75|76**
ADCSFR	**sensorDATA0**	**sensorDATA0**	**sensorDATA1**	**sensorDATA1**	**17**	**77|78**
ADCSFR	**sensorDATA2**	**sensorDATA2**	**sensorDATA3**	**sensorDATA3**	**18**	**79|7A**
ADCSFR	**sensorDATA4**	**sensorDATA4**	**sensorDATA5**	**sensorDATA5**	**19**	**7B|7C**
ADCSFR	**sensorDATA6**	**sensorDATA6**	**sensorDATA7**	**sensorDATA7**	**20**	**7D|7E**
SPI_cfg|SPIDATAI	**SPI_config**	**SPI_config**	**SPI_DI0**	**SPI_DI0**	**21**	**7F|80**
SPIDATAI|SPIDATAO	**SPI_DI0**	**SPI_DI0**	**SPI_DI1**	**SPI_DI1**	**22**	**81|82**
SPIDATAO...]|FE	**SPI_DI1**	**SPI_DI1**	**FE**	**data79**	**23**	**83|84**
Lock|Reserved	LOCK2(96th byte)	reserved0	reserved1	reserved2	24	85|86

Figure 2.18: Dynamic memory structure of the chip

2.5.1.1 UID, lock bytes, and CC

The first 9 bytes contain the double sized UID and the BCC for each level. Bytes 2 and 3 in Block 2 are the so-called static lock bytes. The default setting of these 2 bytes should be 0x00 indicating that all blocks are free to write. Every bit of these 2 bytes indicates the access to a specific block. “0” indicates read and write access and “1” locks a block for read access only. These lock bytes are called static, because there position in memory is fixed and there are always 2 bytes or 16 bits for 16 blocks of the static memory structure, respectively. The third block is made up of the Capability Container (CC) and contains information about the data or data area but not application related data. The first byte contains 0xE1 and indicates that NFC Forum defined data is stored in the data area. It’s also called the “magic number”. The next byte stands for the version number of the NFC Forum Type 2 Tag specification. Byte 2 represents the size of the data area and is 0x0A. The data area size is

$$data\ area\ size = 0xA * 8 = 10 * 8 = 20\ blocks = 80\ bytes.$$

The last byte indicates the read and write access capability of the data and CC area. If the most significant nibble holds the value 0x0, read access is granted without any security. If the least significant nibble holds the value 0x0, write access is granted without any security. If it holds 0xF no write access is granted at all. All other combinations are RFU or proprietary. Here a dynamic memory structure is used, because more than 16 blocks are needed. Therefore an additional third dynamic lock byte is introduced after the data area. Every dynamic lock bit locks 2 blocks, if it's set to "1". According to [5] the number of bits to enable the locking feature for the last 8 bytes of the data area is

$$dynamic\ lock\ bits = \frac{data\ area\ size - 48}{8} = \frac{80-48}{8} = 4.$$

This leads to the number of the lock bytes and is

$$dynamic\ lock\ bytes = \frac{data\ area\ size - 48}{64} = \frac{80-48}{64} = 1\ (rounded)$$

Retrospectively the data area consists of

$$data\ area\ size = 4*(k-3) - dynamic\ lock\ bytes - reserved\ bytes = 4*(24-3)-1-3 = 80.$$

The data area is illustrated in Figure 2.18 with bold text.

2.5.1.2 TLV structures

Most data in an NDEF Message is organized in so-called TLV Structures (Tag-Length-Value). The Tag field codes the type of the structure and the Length is coded in the Length field. One Length field is sufficient, due it is shorter than 254 bytes. If more bytes are necessary, 3 Length bytes must be used. The Value field contains the information [11].

- **Lock Control TLV**

The first TLV in the dynamic memory is the Lock Control TLV, which starts at block-number 4. The type is defined with 0x01 and its Length is 0x03, which indicates 3 bytes following. The coding of the value is illustrated in Figure 2.19:

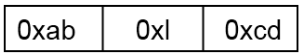

Figure 2.19: Structure of the Lock Control TLV and Memory Control TLV

Adopted from [11]

According to [11] the starting address in bytes is calculated with the following Formula:

$$a * 2^d + b \rightarrow 0xC * 2^3 \rightarrow 12 * 2^3 = 96$$

The meaning of a, b, c, d and l is illustrated in Figure 2.19. The Length is indicated by the value of the second field.

$$l \ = \ 4$$

The locked area (how much bytes can be locked) is calculated with 2^c.

$$locked\ area = 2^3 = 8$$

• **Memory Control TLV**

The Memory Control TLV contains control information about the reserved area. The Tag field holds 0x02 to identify this TLV. The value field is 3 bytes long, so l is 3. The first value byte indicates the start address of the reserved area. The most significant nibble "a" represents the byte address, the least significant nibble "b" codes the byte offset. The second value "l" provides information about the number of reserved bytes and is equal to 3. The most significant nibble of the last Value field indicates the number of bytes per page. The other nibble is RFU. The address is calculated as follows:

$$start\ address \ = \ a \ * \ 2^d \ + \ b \rightarrow C \ * \ 2^3 \ + \ 1 \rightarrow 12 \ * \ 8 \ + \ 1 \ = \ 97$$

• **NDEF Message**

NDEF is an abbreviation for NFC Data Exchange Format and is the standardized data format for NFC Forum Devices. It is identified with a 0x03 in its Tag field. The Length specifies the size of the stored message in bytes, which are 0x42 or 66 bytes respectively (without the terminating byte 0xFE). The NDEF message itself is stored in the Value field. An empty NDEF message TLV holds a 0x00 in the Length field and has no Value field [12].

– **Value field**

An NDEF Message contains one or more NDEF records shown in Figure 2.20. In the case of only one record, it is marked with the Message Begin Flag (MB) and the Message End Flag (ME) set. In the other case the first record has MB and the last record has ME set. An NDEF message can contain many chunked payloads. If the Chunk Flag (CF) is not set, the message does not contain chunked payloads. Short Record (SR) identifies a record with a payload less than 255 bytes. If the ID Flag (IL) is set, the ID Length field and ID field are omitted from record and header. The Type Name Format (TNF) specifies the TYPE field structure. "001" indicates an NFC Forum well-known type [12].

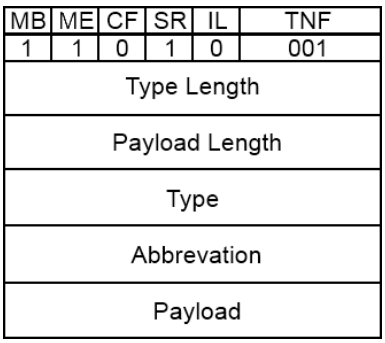

Figure 2.20: A NDEF short record

Adopted from [12]

Type Length specifies the length in bytes of the Type field and in this case its Value is 0x01. The Payload length provides information about the length of the records payload. Type contains the Value 0x55, which identifies a Uniform Resource Identifier (URI) record type ("U"). Abbrevations are used to keep messages short, e.g. 0x03 for "http://" [12].

Here the Value is 0x00, which stands for "no abbreviation". The payload is plain data. Following are 62 bytes of data, which contain several configuration and data words. The terminator 0xFE indicates the end of the NDEF Message. The 79^{th} byte is the last byte of the data area followed by a dynamic lock byte and some reserved bytes.

2.5.2 Command set

The command set of an NFC Forum device consists of the ISO 14443 commands and some data manipulation commands. These are described in Section 2.5.2.2.

2.5.2.1 Select and Anticollision commands

The anticollision mechanism of the ISO 14443 standard is also used in Type 2 NFC Forum Devices. They differ in their naming but have the same function.

2.5.2.2 Read and Write

Additionally 2 commands are used for manipulation and acquisition of stored data. Both frames are terminated by a CRC. For the sake of simplicity no start of communication, parity or end of communication are shown in following figures.

• **READ**

This command requests 16 bytes from the specified Block address, which represents the block where the read cycle should begin. It gets either all requested bytes or a "not acknowledge" (NACK), like illustrated in Figure 2.21 [5].

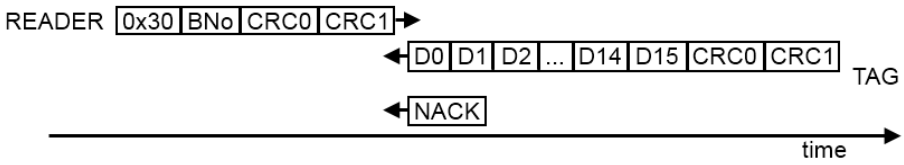

Figure 2.21: Read command

Adopted from [5]

• **WRITE**

A WRITE command consists of 4 bytes which should be stored at a specific Block address. A Tag replies with an "acknowledge" (ACK, all data have been stored successfully) or with an "not acknowledge" (NACK, error has occurred). Figure 2.22 shows the write procedure [5].

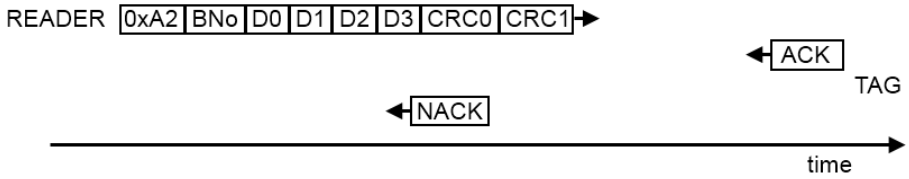

Figure 2.22: Write command

Adopted from [5]

Chapter 3: Component Description

The first Section gives a brief insight into the design process of a Very Large Scale Integrated Circuit (VLSI Circuit). Section 3.2 focuses on some design considerations, which have to be made in order to design a fully functional system. Then the concept of the chip and its components is explained.

3.1 Design Flow

A typical design process of an ASIC is illustrated in Figure 3.1 and is starting with the concept phase. This phase covers design objectives such as desired functionality, chip size, power consumption and other parameters. Designers now shape the components of the system to meet these requirements during the high level chip design phase. Furthermore design testability concepts are developed and discussed.

The desired functionality is implemented in a behavioural model and verified through simulation. The next step is the translation of the behavioural model to an RTL description. This is done in a Hardware Description Language (HDL). The mostly used HDLs are VHDL (Very high speed integrated circiut Hardware Description Language) and Verilog. The behaviour of the RTL Description is simulated and its results are compared with the behavioural model in order to ensure translation success. Until now the design is completely technology independent.

The last step before fabrication consists of synthesizing the RTL model to a particular technology. Usually this step is performed automatically by a synthesis tool. In order to ensure correct behaviour, this Structural Hardware Description is simulated and compared to the RTL Description. The design is either implemented as Application Specific Integrated Circuit (ASIC) or as a PLD (Programmable Logic Device) solution [7].

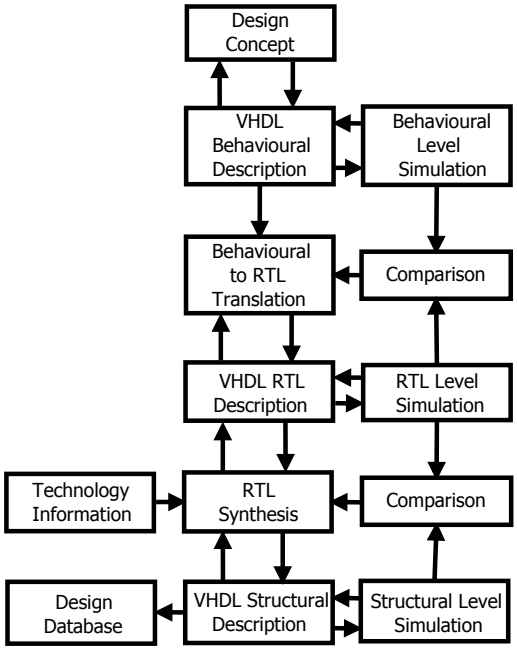

Figure 3.1: Design flow for designing VLSI circuits

Adopted from [7]

3.2 Design Considerations

This Section presents a brief look on some design considerations. At first it will focus on general problems which can occur during digital design. Afterwards the handling with Multiple Clock Domains is discussed.

3.2.1 General Thoughts

The first consideration to be discussed is the chip area. The design should be kept small due to minimal use of registers. Therefore it should be specified how much registers or memory elements are really necessary in order to minimize the silicon footprint. The required power consumption is another important point. The best opportunity to optimize

the power consumption is the so called clock gating. Clock gating ensures that currently unused circuit blocks are not provided with a clock source. If the clock of a Flip-Flop is deactivated it avoids unnecessary switching. This saves a great amount of power, because in CMOS technology the dynamic power dissipation (during switching states) is much higher than the leakage. Clock gating can be achieved in RTL design by adding enable conditions to blocks or units [10]. A digital design is always a tradeoff of performance, area and power consumption. It has to be specified which of these points has highest priority. Because to enhance the performance parallel computing (more area) or higher frequencies are a possible solution. These points should be kept in mind during a digital design.

3.2.1.1 Loops

This problem occurs in combinational designs. The combinational function in Figure 3.2 makes reference to its own result. In this zero-latency or combinational loop it is not possible to determine a logic value for the output even if the input values are known[10]. Such contructs can be avoided if the output in a combinational design does not refer to its input. There are rare situations where such constructs are usable.

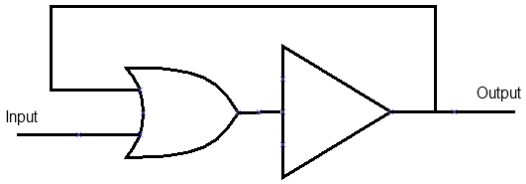

Figure 3.2: Combinational loop

3.2.1.2 Latches

A design can be eiter sequential or combinational. In combinational designs the output depends only on the inputs. Sequential designs contain memory elements (e.g. Flip-Flops). Then the output depends not only on the actual input but also on the condition of earlier inputs. A clock triggered Flip-Flop stores the data at its input if a clock edge is detected. A Latch stores the data if the clock (or enable input) is at high state [10]. Verification, testing and synthesis are much easier to perform if the design contains only edge triggered digital components such as Flip-Flops. Many synthesis tools support only designs with Flip-Flops. In that case the designer has to take care that the Latch input does not changes nearly at the same time as the enable input of the Latch. This can lead to

unexpected behaviour. The Latch may mix old and new data and store this mixture. Therefore Latches should be avoided. This problem is not existent if only Flip-Flops are used. But also Flip-Flops are prone to other problems like metastability. This is explained in the next section.

3.2.1.3 Synchronisation and Multiple Clock Domains

A very common mistake in digital designs may be a proper treatment of asynchronous inputs (as in systems with Multiple Clock Domains).
The input of a Flip-Flop must be stable for a specified time before and after the triggering clock edge. This setup time and hold time is illustrated in Figure 3.3 as t_{su} and t_{ho}. While in the first case timing requirements are respected, data changes during the data-call window in the last 2 cases, which causes metastable behaviour. It can resolve in time to a stable state or not. In any case it was empirically determined that the metastability resolution time outruns the propagation delay t_{pd} by orders of magnitude. The unpredictable delay is the main problem and not the random outcome [10].

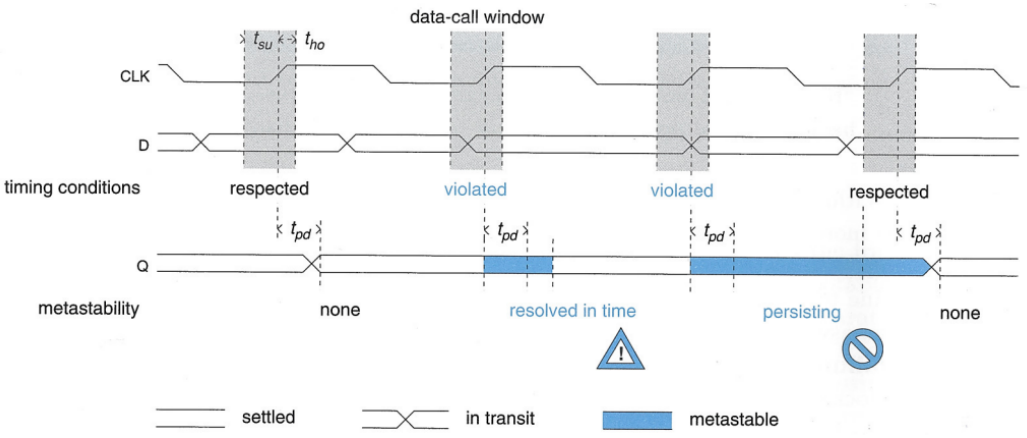

Figure 3.3: Reasons for metastable behaviour

Adopted from [10]

A single stage synchronizer is shown in Figure 3.4. The Flip-Flop (FF) is negative clock edge triggered and so adopts the asynchronous data X(H) at the next negative clock edge. The signal has now enough time to resolve to a stable state, until it is adopted by the FSM component, assuming the setup and hold times are much smaller than a clock period. Yet input changes of the synchronizer itself during the sampling interval may lead to errorenous behaviour too. Thus it could pass the metastabile state on to the

FSM [16]. According to [10] the Mean Time Between Errors (MTBE) for a single stage synchronizer is determined by

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

with

$$t_{al} = T_{clk} - max(t_{pdc}) - t_{suffdnst}.$$

where f_{clk} is the clock frequency and f_d the data input change frequency. Values of K_1 and K_2 depend on the used technology and electrical characteristics of the Flip-Flop. t_{pdc} is the propagation delay of the longest path and $t_{suffdnst}$ stands for the setup time of the downstream Flip-Flop.

MTBE gets higher with a lower clock frequency. For the frequency of 1.92 MHz the Mean Time Between Failures (MTBF) is extremely high, because $t_{al} \geq 3t_{pd}$. Still a two stage synchronizer is used. At usage of a cascade of two Flip-Flops the MTBE is squared [10].

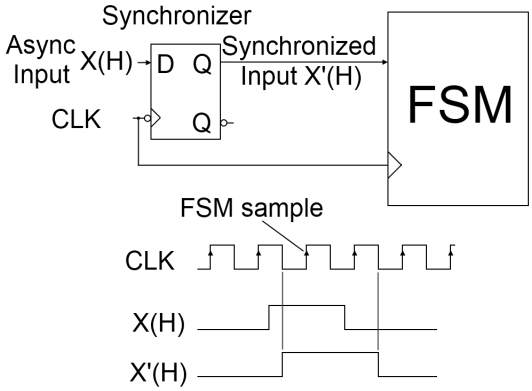

Figure 3.4: Single Stage Synchronizer

Adopted from [16]

Individual signals are synchronized with a two stage synchronizer. If data applied to a bus have to be be synchronized, another solution needs to be found. The easiest way is the four handshake procedure, shown in Figure 3.5. At first data is applied to the bus and the synchronized valid signal is set. This data must not change until a synchronized acknowledge is received. Then data is free for change and the valid signal will be reset. This will guarantee that these signals are set long enough to be captured, regardless of the direction (slow to fast clock or fast to slow clock).

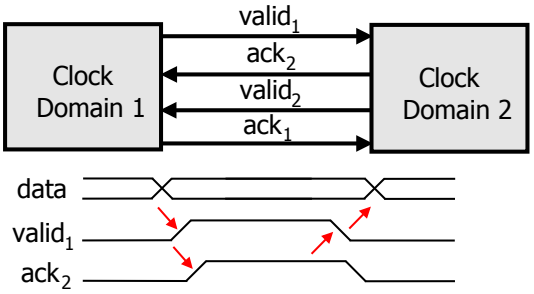

Figure 3.5: Synchronizing scheme

3.3 Concept

The task was to develop a digital communication part for an existing digital frontend, which already supports the EPCglobal Class-1 Gen2 Standard. The chip should be extended in that way to also support the NFC standard for communication. Furthermore it should have a low power consumption and minimal silicon footprint. In [3] the previous version of the chip is discussed.

This chapter starts with a concept, which gives an overall view about the new chip structure. It includes analog as well as digital components. Later on the focus will switch to digital only. At first the new version of the chip will be described. The parts themselves and how they work together with the new digital frontend will be discussed in more detail. The concept is illustrated in Figure 3.6. Colored blocks represent digital components, while analog units are blank.

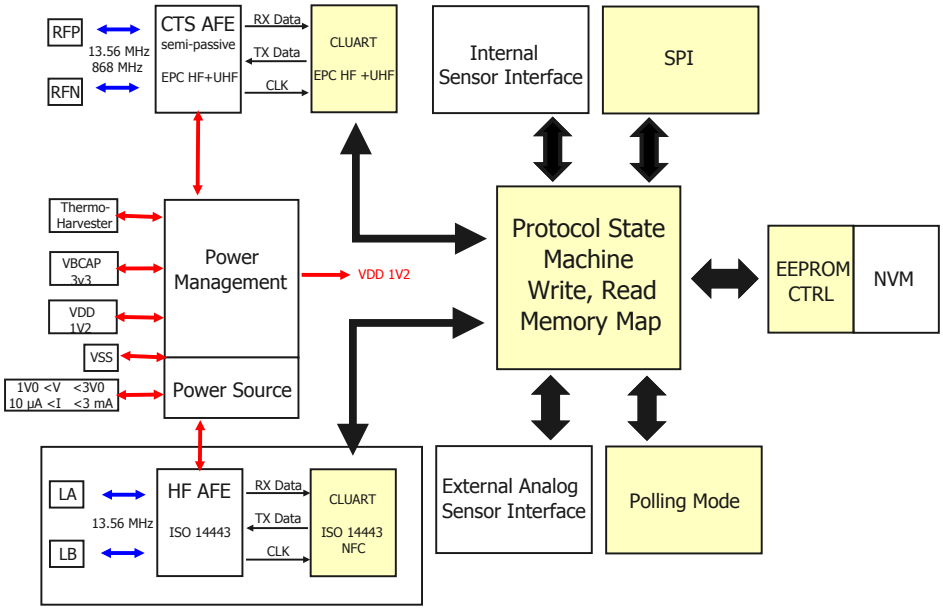

Figure 3.6: Chip concept

The Power Management Block provides the system with the required power. It is supplied either from an external source (Reader) or by using a built-in Thermo-Harvesting circuitry. However the external source is either a CTS AFE or an ISO 14443 AFE. These serve

as the clock source and provide each CLUART with data through the antenna. In this case they convert the signal received from the RF Field into bits and bytes and vice versa. This data is used to execute commands. Sensors are controlled by specific data patterns within a WRITE command. The internal sensors consist of an Analog to Digital Converter (ADC) and can be used, for example, to measure the actual environment temperature. In order to achieve this, data has to be written to a specific address. The data can then be read out from sensors, the Serial Peripheral Interface (SPI) or the NVM by a READ command. WRITE commands can either be used to write data to NVM, SPI or to activate sensors. In the case of communication via SPI, the Power Management Block is supplied through the VBCAP, which is a capacitance, that works as battery. The Polling mode is started via a WRITE command to a specific address and measures the actual temperature regularly. The interval is also adjustable by this command. If the Reader requests the data, the ADC contains the highest temperature measured during the polling period. The Polling unit was designed independently of the NPU.

3.4 Chip structure

The main work is to design the NFC Protocol Unit (NPU) and to integrate it into the existing bus system. It will perform as a bus master unit, such as CTS and the SPI. Figure 3.7 shows the basic digital core components of the system. The additional component has to communicate via the internal bus in order to be able to control the slave units. The next sections will explain the units and their responsibilities within the system.

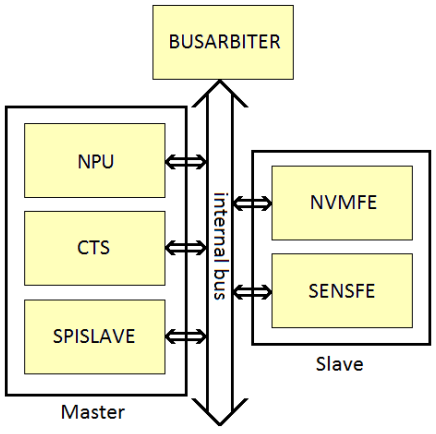

Figure 3.7: Digital Top Unit

3.4.1 Bus Arbiter

This bus is designed as a multimaster and multislave system. The Bus Arbiter is responsible for providing master status at the bus. Every bus master unit communicates with the Bus Arbiter via a “bus request “ “bus granted“ and “slave ready“ signal. In order to achieve master status, the unit has to apply the data, and command (read or write) to the bus. Then it has to request master status. The Bus Arbiter recognizes a request and the unit obtains master status if the bus is free. Therefore the Bus Arbiter sets the “bus granted“ signal. Now the master can control datatransition via the bus, for example requesting data from a slave. Slaves communicate via an “activate“ “acknowledge“ and “ready“ signal with the Bus Arbiter. In the case of a master request, the Bus Arbiter activates the slave with the corresponding signal. The slave acknowledges the request and reads the data applied to the bus by the master. After it has finished (e.g. sensor data is ready), the “ready“ signal is forwarded to the master unit by the Bus Arbiter.

3.4.2 NPU

This block is the new digital communication frontend and it is responsible for communication via NFC. It decodes the incoming stream and interprets and executes the readers commands.
The NPU will be explained later in more detail. The functionality of the other blocks will be discussed as far as it’s necessary to understand their cooperation with the NPU.

3.4.3 CTS

The CTS unit is responsible for handling the communication via the EPCglobal Class-1 Gen2 protocol. It decodes the incoming stream and executes the readers commands.

3.4.4 SPI

The third bus master is the SPI. It converts the incoming serial bit stream into the bus format and also executes incoming commands.

3.4.5 NVMFE / SENSFE

A slave cannot initiate a bus transfer itself but it can respond to a communication request by one of the bus masters. Then it provides access to the memory (NVM) or sensors.

3.4.6 Memory structure

The EPC memory structure is illustrated in Figure 3.8. It consists of four banks. For this project only the userbank (Bank 11) is available for use. Detailed information about the rest of the memory and its functioning is available in the EPCglobal Class-1 Gen2 Standard documentation [9]. The used memory starts at address 0x55 with the UID and goes on with the NDEF Message. It consits of sensor and SPI configuration bits and the sensordata or adc data. SPI data in / data out bytes are placed before the terminator byte and the reserved bytes of the NDEF Message, as illustrated in Figure 2.18.

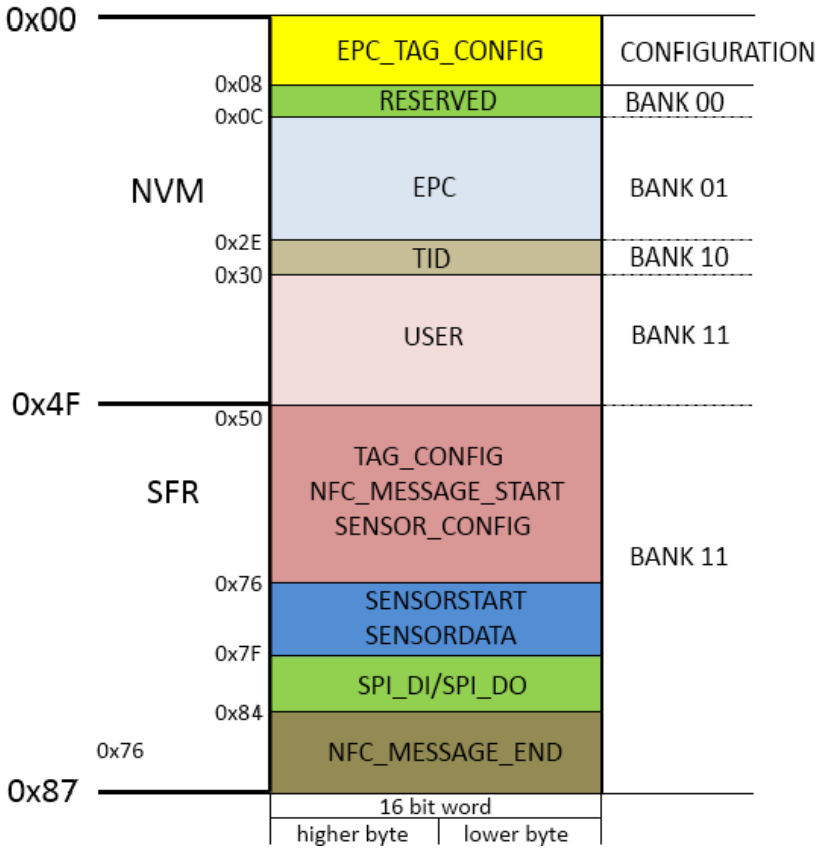

Figure 3.8: EPC Global Memory organisation

3.5 RISC Microcontroller and added Peripherals

As mentioned before, one of the tasks was the integration of the NFC Communication Frontend into an existing system with an own bus protocol. At first it was necessary to develop a component, which understands the NFC Protocol. For this reason, a RISC controller (hereinafter referred to as RISC) is used. It was specifically designed for RFID applications including a very small silicon footprint. A small but powerful Instruction Set enables it to fulfill the task easiliy. The instructions are stored in a Read Only Memory (ROM). Furthermore programming with this Instruction Set is straight forward. This allows a high optimization for power consumption, area and performance.

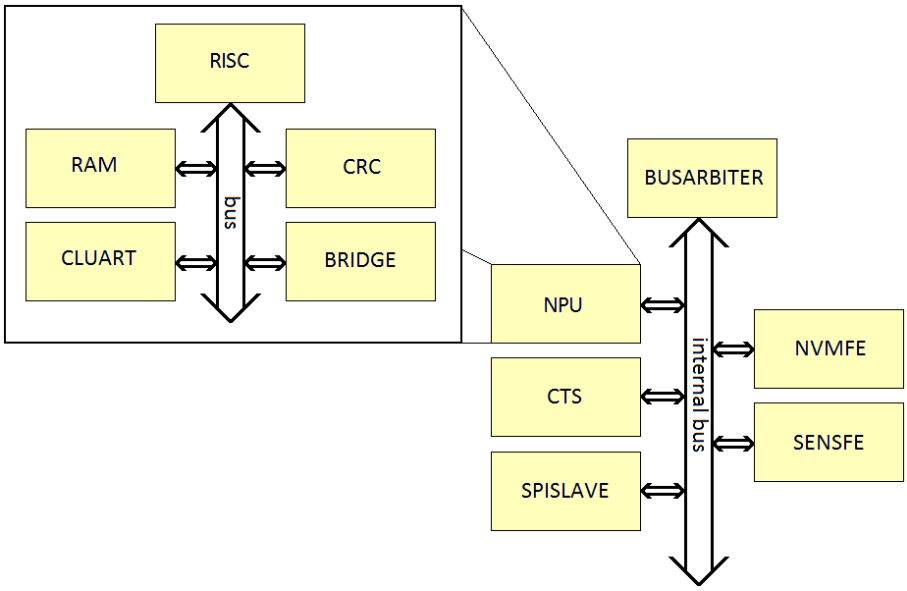

Figure 3.9: DTU with embedded RISC and Peripherals

It uses a specific bus protocol, where it's possible to attach several peripherals. All attached peripherals are illustrated in Figure 3.9 and they will be discussed now in detail. Error handling is also implemented but it is not illustrated in all flow diagrams. This ensures a better understanding. According to the protocols requirements, proper error handling is done in case of communication errors.

3.5.1 RISC

This RISC offers a small instruction set in order to meet the requirements of a typical RFID product. In Figure 3.10 the inputs and outputs are shown. Ports with an "_i" at the end represent inputs, while ports with an "_o" represent outputs. Busses are represented with thick arrows. In addition to the obligatory clock input also an asynchrounous reset is necessary to reset the circuit into a predetermined start state.

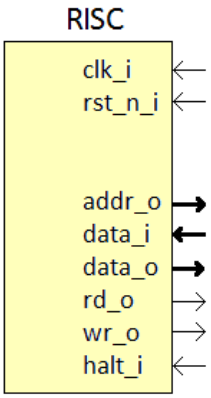

Figure 3.10: Ports of the RISC

In order to communicate with other peripherals an address bus and a data bus are implemented, which both have a bus width of 8 bit. Also it can inform the peripheral whether it wishes to read or write data from it. The peripheral, by contrast, is allowed to freeze the RISCs operation, if it's not able to react to its demand immediately and to release it, when the peripheral is ready to respond. So no interrupt functionality is implemented.

3.5.1.1 Communication between the RISC and Peripherals

Figure 3.11 shows the handshaking mechanism, which is used by the RISC and its peripherals. In this system the RISC is the master and determines the data traffic on the bus. All data has to be processed by it.

Peripherals are supplied with the same clock and are reset at the same time as the RISC. If it's known that data will be provided by the Reader with the help of the peripheral CLUART, the RISC has to request data from it. The peripheral itself is not allowed to control the dataflow on the bus. Therefore the RISC sets its read output (rd_o) to

high and specifies the appropriate address of the peripherals special function register (SFR). These outputs are read by all components which are attached to the RISC, but the address determines which component replies to the read command. For example if CLUART isn't ready yet, it can set its output halt_o to high to inform the RISC that no data is available yet. After CLUART has finished the data acquisition, it releases the RISC by forcing halt_o to low and simultanously providing the required data to the bus.

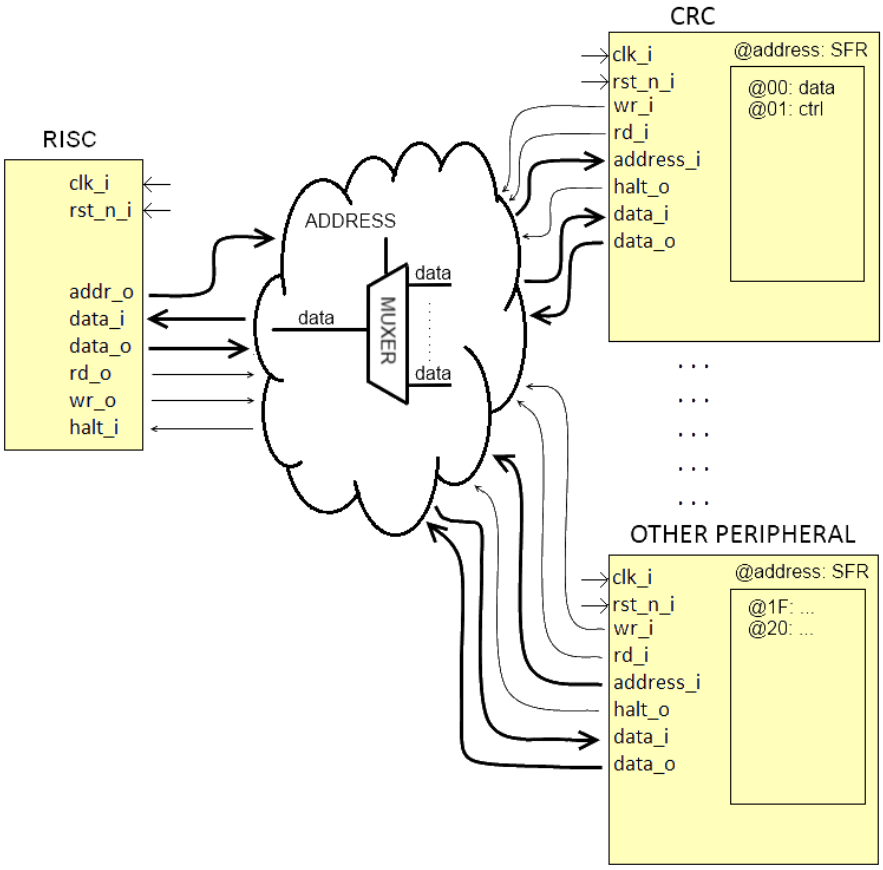

Figure 3.11: Comunication between the RISC and its peripherals

This handshaking mechanism is used in order to communicate with all peripherals. In the case of writing data the mechanism differs only in using wr_o instead of rd_o. Every peripheral has a number of SFRs which can be used for data transportation, configuration or indication of the status of the component. Since it has an address bus width of 8 bit, there are a maximum of 256 SFRs respectively peripherals addressable.

3.5.1.2 Anticollision flow diagram

Figure 3.12 shows a simplified flow diagram for the RISC, which doesn't include exact commands or data transfers between registers. It should only explain how the anticollision process is handled. It represents the implementation of the anticollision process by using the RISC.

After a PICC enters an RF field, it powers up. Immediately the UID is transferred from NVM to the Random Access Memory (RAM), allowing easier access to it during the anticollision process. After that the PICC waits for a REQA or WUPA command. If one of these commands is received the RISC reaches READY state after responding with an ATQA. Now the anticollision loop starts. This means the RISC reacts only, if the next command is a select command beginning with 0x93. If the following byte is a 0x20, it responds with the whole first level UID and the BCC. The calculated CRC is rejected, because it is of no use until now. At this time, if there appear more than one PICCs in the field, the PCD detects a collision. As described in chapter 2, it sends another select command, but now followed by the number of bits and bytes, which were detected before the collision. This is neither 0x20 nor 0x70.

The RAM is used to compare the received bytes in a loop, whichs counter is generated out of the information of the NVB. The RAM stores the data byte wise, so for comparing a splitbyte, the last byte which was read out of the RAM has to be masked in such way that it consists out of as much bits as received. After this was finished and all received bytes were compared and correct, the PICC responds by sending out the remaining bits and bytes terminated by a BCC.

This procedure will be performed many times, depending on how much PICCs operate within the RF field and in what extent their UIDs differ. If at least an individual PICC was determined the PCD sends out the last select command (0x93 for level 1).An NVB follows, which stores 0x70 then the UID and the calculated CRC. If the received data copes with the data from the RAM and the Cyclic Redundancy Check passes, then the PICC responds with a level 0 SAK and a CRC. Then the second anticollison loop starts with 0x95 as select command. Now the anticollision procedure is repeated in the next UID cascade level. The SAK defined for the second level finishes the anticollision sequence and the PICC enters ACTIVE state. READ and WRITE will be explained next.

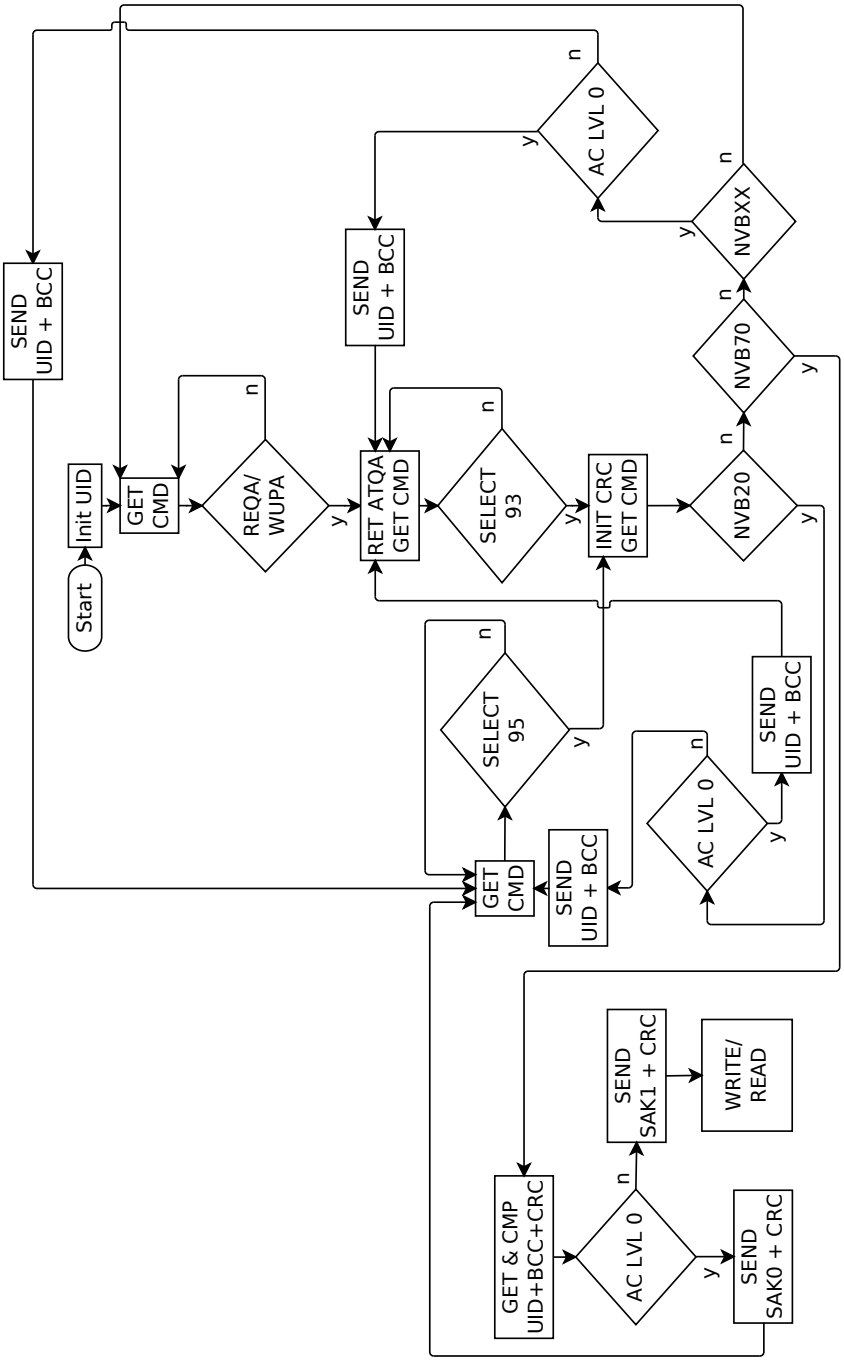

Figure 3.12: Simplified anticollision flow diagram

3.5.1.3 Read/Write flow diagram

Figure 3.12 shows the procedure, how a PICC is selected out of many PICCs operating in the RF field. Figure 3.13 illustrates the READ and WRITE procedure. After selection the PICC is waiting for further commands. These commands are either READ, WRITE or HLTA. If the PICC receives a command, it starts the CRC calculation component. Afterwards it needs to be determined whether a READ or WRITE command was received. This is done by checking the first received byte. It should be eiter 0x30 (READ command) or 0xA2 (WRITE command). If the RISC requests data, it first loads the words consecutively from the specified address (requesting from SFR, NVM, SPI or sensors). Then it stores the words, each separated into 2 bytes, into the RAM until 8 words or 16 bytes were transferred. When all data is stored, the CRC is initialized and the data is sent to the PCD. The CRC is calculated over all 16 bytes and is attached to them. The RISC remains in ACTIVE state, where it reacts only to HLTA, READ and WRITE commands. If another command is received an error is detected and it returns to IDLE or HALT state. If the PCD desires to write data the RISC gathers all data to be written (4 bytes) and stores it into the RAM. After that the RISC delivers the data to the bridge and waits for an acknowledge. This informs it, that data has been successfully stored. Then the RISC transmits an ACK to the PCD. If an error occured NACK is sent to the PCD. Here the RISC remains in active status too.

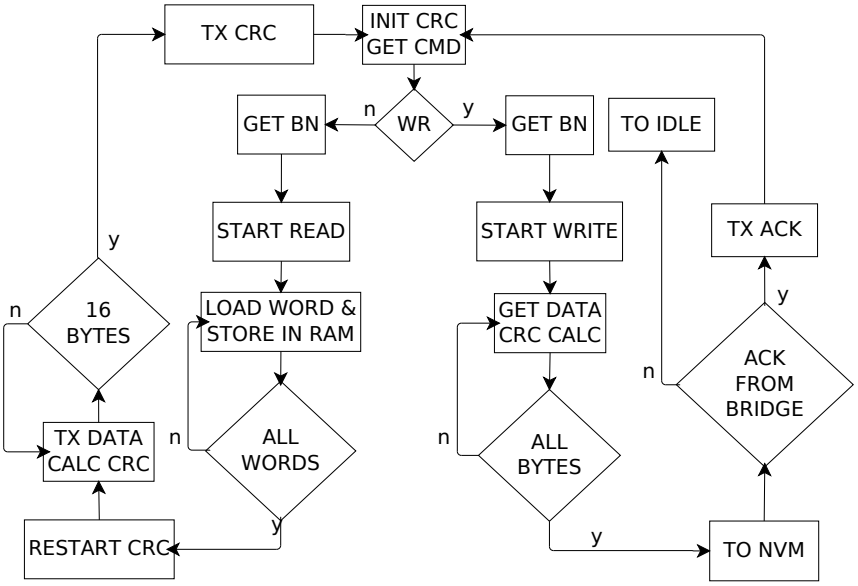

Figure 3.13: Simplified flow diagram during ACTIVE state

3.5.2 Peripherals

This section explains the function of all peripherals attached to the bus system of the RISC. These are CLUART, RAM, CRC and the BRIDGE.

3.5.2.1 CLUART

The CLUART is not described in detail, because it's intellectual property of Infineon Technologies AG. But the main functionality is the serial/parallel data conversion and data encoding and decoding to enable communication with the PCD. It delivers the data byte wise to the RISC. Furthermore it receives the data from the RISC byte wise and ensures proper timing to meet the requirements introduced in Section 2.4.

3.5.2.2 RAM

The RAM is used to store the PICCs UID if it enters an RF field. Furthermore it buffers data, which the RISC gets during the read and write procedure. There are 32 8 bit registers. Ten of them hold the Cascade Tag, UID and BCC. The remaining are free to use. Addressing is rather simple, because the content is stored in 8 bit registers.

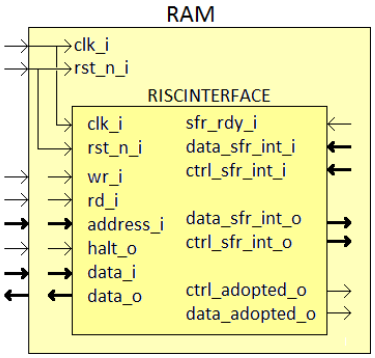

Figure 3.14: Ports of the external RAM component

Figure 3.14 illustrates the inputs and outputs of the RAM peripheral. It is noticeable that there is another component instantiated. This component is responsible for the dataflow between the RISC and the RAM. Its SFR inputs and outputs are connected to registers within the RAM. The CTRL_SFR has the address 0x07 and the DATA_SFR has the address 0x06. Their function is shown in Figure 3.15.

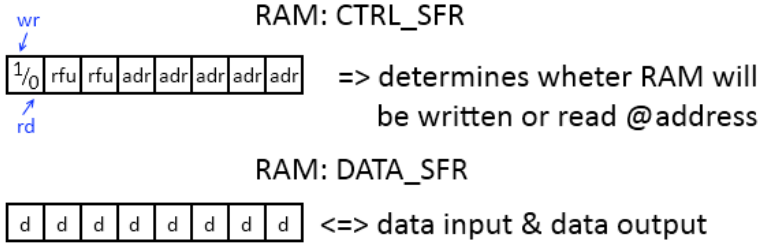

Figure 3.15: Special Function Registers of the external RAM component

If an SFR is manipulated by a component, it informs the RISC by setting sfr_rdy_i to "1". Due to program flow the RISC knows, which SFR has been manipulated. The RISCINTERFACE then forwards the SFR to the RISC. Conversely the component is informed about a new SFR with ctrl_adopted_o and data_adopted_o. This information flow has the same structure in all following components.

3.5.2.3 CRC

The CRC unit uses the same interface as the RAM to communicate with the RISC. The purpose of the CRC unit is the calculation of the checksum during reception and transmission of data. Address 8 represents the DATA_SFR and address 9 the CTRL_SFR. Therefore it receives a byte of data, which has to be computed by the CRC algorithm. This algorithm is the CRC-16/CCITT generator polynomial and consits of 2 bytes. According to [4], it has the following structure:

$$x^{16} + x^{12} + x^5 + 1$$

with the initial value of 0x6363.

Another view on this algorithm is presented in Figure 3.16 where it's implemented with simple shift registers. This implementation method guarantees minimal chipspace. With the use of a CRC mechanism a high data integrity is achieved [4].

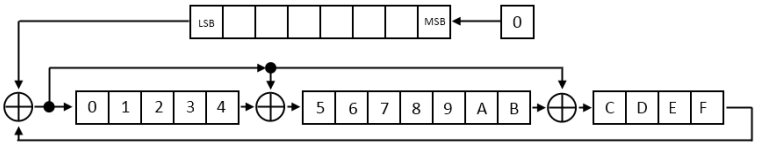

Figure 3.16: CRC value calculation with shift registers

Adopted from [4]

The 2 CRC bytes are calculated during reception and transmission and are attached at the end of the data frame. If a data frame already containing a CRC is received then the calculation includes these two CRC bytes and the result should be 0x00, if no error has occurred.

DATA_SFR is again responsible for data transfer. At the beginning the RISC stores 0xF0 into the CTRL_SFR, which signals the CRC peripheral, that all following data bytes are input for the CRC calculation algorithm. However, if the RISC sets the value of CTRL_SFR to 0xFF, this signals the peripheral, that calculation has finished. As there are 2 bytes of a CRC value, the RISC has to request both bytes after delivering the last byte for calculation. It also signals that no further bytes will be provided with setting CTRL_SFR to 0xFF. The SFRs and their function are shown in Figure 3.17.

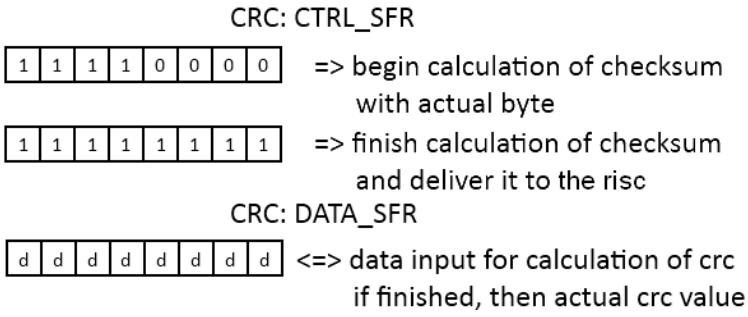

Figure 3.17: Special Function Registers of the CRC unit

3.5.2.4 BRIDGE

The main function of this component is to map the commands and data received from the PCD to the internal bus interface. The left side, as shown in Figure 3.18, is the interface to the RISC bus, running with 848 kHz while the right side represents the interface to the chips internal bus with the Bus Master. Here the clock source is 1.92 MHz. Thus the bridge is a multiple clock domain. In order to ensure data integrity clock synchronization techniques as described in Section 3.2.1.3 have to be used.

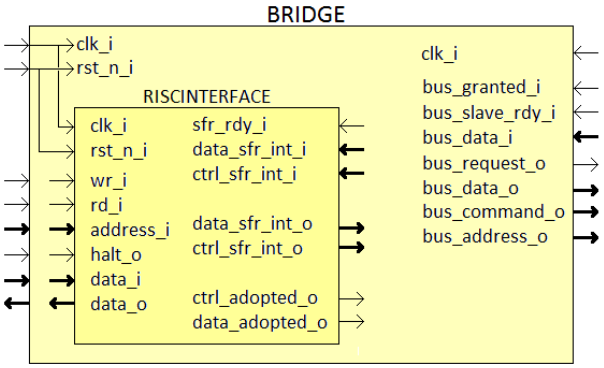

Figure 3.18: Ports of the bridge component

In Figure 3.8 it can be seen that the used NVM consists of 16 bit blocks, while the NFC Forum compliant memory should be accessed blockwise with 4 bytes per block. As the RISC works with an 8 bit bus, it's the best idea to take 8 bit as lowest common denominator. So every two bytes from the RISC are merged to a 16 bit word, with respect to the significance of each byte, and vice versa. Figure 3.19 gives a practical view of the data mapping. The left side represents the NFC memory layout, while the right side shows the EPC memory layout.

block 0 of the nfc memory layout with a dimension of 32 bit corresponds to page 55 and 56 of the epc memory layout with a dimension of 16 bits

	ByteNumber				BNo	Addr
Content	0	1	2	3	dec	hex
UID|BCC0	UID0	UID1	UID2	BCC0	0	55|56
UID_2nd_part	UID3	UID4	UID5	UID6	1	57|58
BCC1|Int|Lock	BCC1	Internal2	Lock0=0x00	Lock1=0x00	2	59|5A
CC	CC0=0xE1	CC1=0x10	CC2=0x0A	CC3=0x00	3	5B|5C
LockControlTLV	**LCTLV0=0x01**	**LCTLV1=0x03**	**LCTLV2=0xC0**	**LCTLV3=0x04**	**4**	**5D|5E**
LockCtrlTLV|MCTLV	**LCTLV4=0x33**	**MCTLV0=0x02**	**MCTLV1=0x03**	**MCTLV2=0xC1**	**5**	**5F|60**
MCTLV|NDEFMsgTL	**MCTLV3=0x03**	**MCTLV4=0x03**	**NDEFT=0x03**	**NDEFL=0x42**	**6**	**61|62**
NDEFV	**NDEFV0=0xD1**	**NDEFV1=0x01**	**NDEFV2=0x3E**	**NDEFV3=0x55**	**7**	**63|64**
NDEFV|NDEFMsg	**NDEFV4=0x00**	**NDEFSKIPNULL**	**SENSOR_CONFIG00**	**SENSOR_CONFIG00**	**8**	**65|66**
NDEFMsg	**SENSOR_CONFIG01**	**SENSOR_CONFIG01**	**SENSOR_CONFIG02**	**SENSOR_CONFIG02**	**9**	**67|68**
NDEFMsg(config)	**SENSOR_CONFIG03**	**SENSOR_CONFIG03**	**SENSOR_CONFIG04**	**SENSOR_CONFIG04**	**10**	**69|6A**
NDEFMsg(config)	**SENSOR_CONFIG05**	**SENSOR_CONFIG05**	**SENSOR_CONFIG06**	**SENSOR_CONFIG06**	**11**	**6B|6C**
NDEFMsg(config)	**SENSOR_CONFIG07**	**SENSOR_CONFIG07**	**SENSOR_CONFIG08**	**SENSOR_CONFIG08**	**12**	**6D|6E**
NDEFMsg(config)	**SENSOR_CONFIG09**	**SENSOR_CONFIG09**	**SENSOR_CONFIG10**	**SENSOR_CONFIG10**	**13**	**6F|70**
NDEFMsg(config)	**SENSOR_CONFIG11**	**SENSOR_CONFIG11**	**SENSOR_CONFIG12**	**SENSOR_CONFIG12**	**14**	**71|72**
NDEFMsg(config)	**SENSOR_CONFIG13**	**SENSOR_CONFIG13**	**SENSOR_CONFIG14**	**SENSOR_CONFIG14**	**15**	**73|74**
NDEFMsg(config)	**SENSOR_CONFIG15**	**SENSOR_CONFIG15**	**sensorSTART**	**sensorSTART**	**16**	**75|76**
ADCSFR	**sensorDATA0**	**sensorDATA0**	**sensorDATA1**	**sensorDATA1**	**17**	**77|78**
ADCSFR	**sensorDATA2**	**sensorDATA2**	**sensorDATA3**	**sensorDATA3**	**18**	**79|7A**
ADCSFR	**sensorDATA4**	**sensorDATA4**	**sensorDATA5**	**sensorDATA5**	**19**	**7B|7C**
ADCSFR	**sensorDATA6**	**sensorDATA6**	**sensorDATA7**	**sensorDATA7**	**20**	**7D|7E**
SPI_cfg|SPIDATAI	**SPI_config**	**SPI_config**	**SPI_DI0**	**SPI_DI0**	**21**	**7F|80**
SPIDATAI|SPIDATAO	**SPI_DI0**	**SPI_DI0**	**SPI_DI1**	**SPI_DI1**	**22**	**81|82**
SPIDATAO/FE	**SPI_DI1**	**SPI_DI1**	**FE**	**data79**	**23**	**83|84**
Lock|Reserved	LOCK2(96th byte)	reserved0	reserved1	reserved2	24	85|86

Addr	BNo	Content
hex	hex	
55	0	UID1|UID0
56	0	BCC0|UID2
57	1	UID4|UID3
58	1	UID6|UID5
59	2	INT2|BCC1
5A	2	LOCK1|LOCK0
5B	3	CC1|CC0
5C	3	CC3|CC2
5D	4	LC1|LC0
5E	4	LC3|LC2
5F	5	MC0|LC4
60	5	MC2|MC1
61	6	MC4|MC3
62	6	L|T
63	7	TYPE_LEN|REC_HEAD
64	7	URI_TYPE|PAY_LEN
65	8	NULLSKIP|URI_H_ID
66	8	SENSOR_CONFIG00
...	...	...
75	10	SENSOR_CONFIG15
76	10	sensorSTART
...	...	...
7E	14	sensorDATA7
7F	15	SPI_CONFIG
80	15	SPI_DI0
81	16	SPI_DI1
82	16	SPI_DO0
83	17	SPI_DO1
84	17	FE|data79
86	18	LOCK2|res0
87	18	res1|res2

Figure 3.19: Mapping of memory from NFC to EPC

In this version only SFRs are used. In other words, it is not possible to save data on a long-term basis, because NVM isn't implemented. But it is possible to change the addressing to support the NVM. For now it is important to test the functionality, so NVM isn't necessary yet. CTRL_SFR has the function to determine whether data has to be written or read.

The SFRs for the Bridge are illustrated in Figure 3.20. Byte 0x20 informs the bridge about the fact, that the RISC desires to read a byte from the specified address. This is done eight times for all words respectively 16 bytes. After storing these into RAM, the job is acknowledged with a 0xFF. The RISC signals the end of the communication process with a 0xF0. Data is transferred via the DATA_SFR.

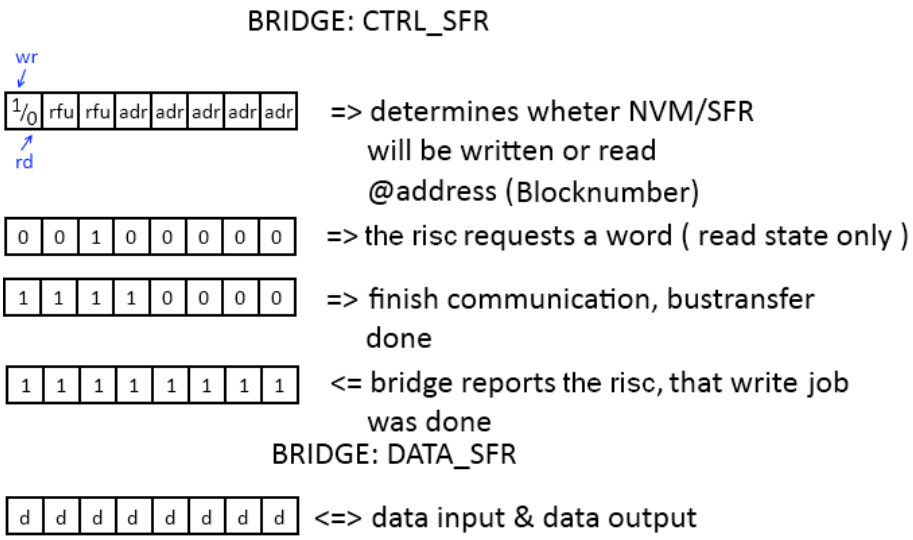

Figure 3.20: Special Function Registers of the Bridge unit

- **State machine**

The bridge processes the data transfer between the bus of the RISC and the internal system bus by a Finite State Machine (FSM). The state changes are clocked with 848 kHz. Figure 3.21 shows a state diagram with all corresponding states and transitions. These will be discussed now.

– **GET_CMD**

After a reset the statemachine is initialized in the GET_CMD state. It waits for either a READ or a WRITE command, which has to be delivered by the RISC. It changes its state to GET_ADD_INFO only, if ctrl_adopted_i is set. The RISC delivers the command and the address with the CTRL_SFR byte. This address corresponds to the Blocknumber used in the NFC protocol.

– **GET_ADD_INFO**

The MSB of the CTRL_SFR determines the further process. If a WRITE command has to be performed the Bridge also gathers the following 4 bytes. These are delivered via the DATA_SFR byte. The internal state (int_state) signals when all bytes were received. In the case of a READ command the state is changed during the next clock cycle.

– **START_WRITE / START_READ**

Here the address (Blocknumber) is read out from the CTRL_SFR. In background the corresponding addresses for a mapping to the EPC address space are calculated. Two addresses are needed for 4 bytes, if data has to be written. 8 addresses are calculated during a READ command.

– **WRITE_NVM / WAIT_WRITE_RDY**

In this state a loop is started, which writes data to memory, sensors or SPI. In WRITE_NVM the bridge requests bus master status at the bus. When it has been granted the bridge applies the internal bus command to signal that data has to be written. Additionally the data word (2 bytes) and the mapped EPC address are applied to the bus along with the synchronized valid signal for synchronization purposes. The bridge changes to WAIT_WRITE_RDY after a synchronized acknowledge. There it waits for a synchronized ready signal from the bus slave.

Half of the data has been stored now. After switching back to WRITE_NVM state it applies the other word to the bus. If the bus slave has dealt with the data, the WRITE procedure has finished and two words are stored in the memory or the sensor is activated. Then the FSM changes its state back to the GET_CMD state. The internal state determines wheter all bytes were written.

– **READ_NVM / WAIT_RD_RDY / NVM_TO_RISC**

The bridge applies the command and the address to the bus and requests bus master status. Now in WAIT_RD_READY the bridge waits for granted master status access and changes its state to READ_NVM. After receiving a ready signal from the bus slave the word has been read out from memory. The bridge withdraws its request and changes to NVM_TO_RISC. Depending on the internal read state the bridge applies the two data bytes to the RISC Bus, if it is requesting data. When the internal read state switches to FINISH the whole process is repeated starting in the state READ_NVM. This is done until all bytes are transferred into the external RAM. This is signaled by the RISC by storing 0xF0 into the CTRL_SFR. Then the bridge switches back to GET_CMD state. During the read loop a CTRL_SFR of 0x20 informs the bridge, that the RISC requests the next data word.

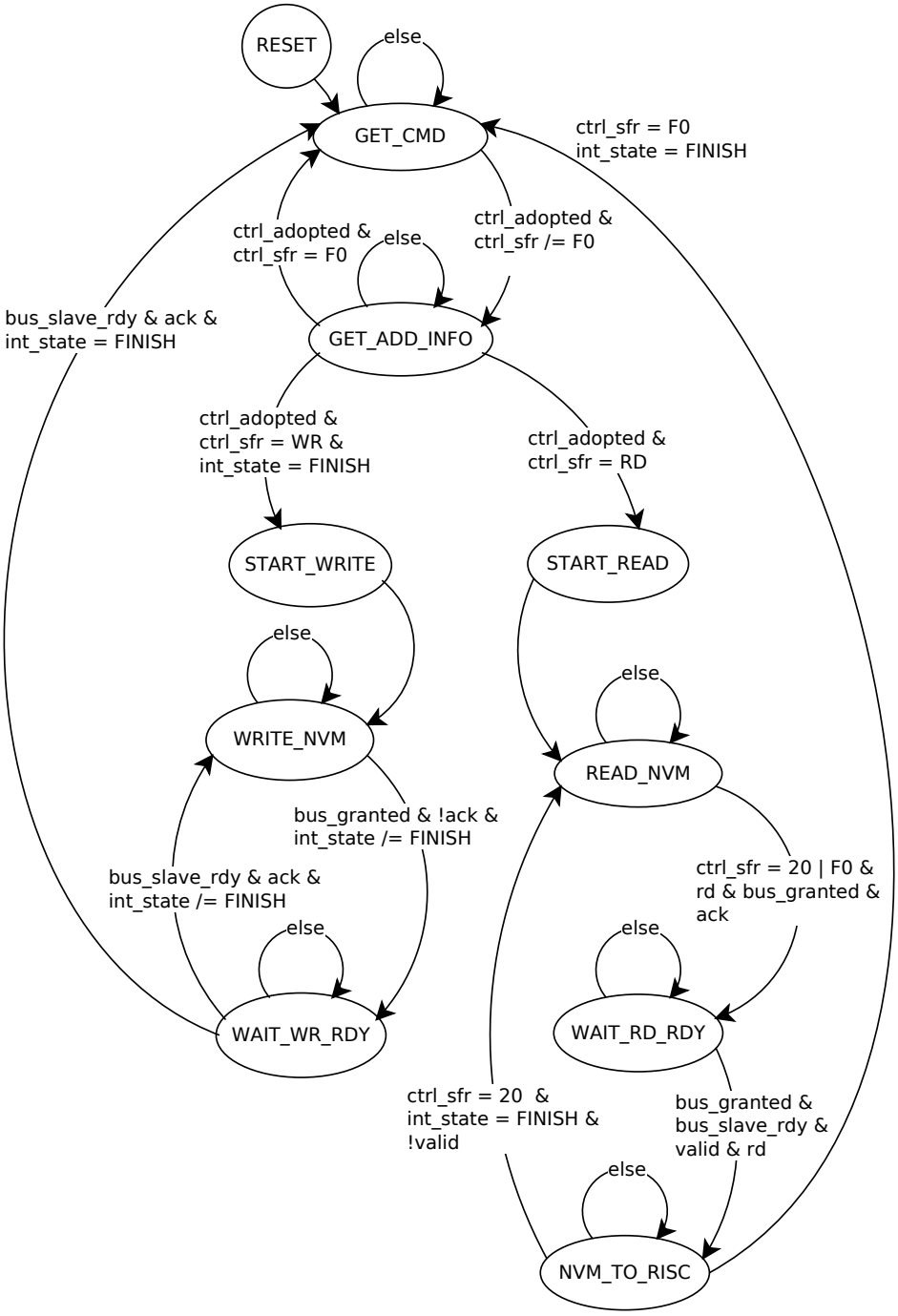

Figure 3.21: State Diagram of the Bridge Data Access

3.5.3 Polling Component

For the Polling Mode, as shown in Figure 3.6, another module was developed independently of the NPU. This component is a prototype for measuring temperature over time. In order to start the polling, the Reader transmits a WRITE command to a specific address. When enabled, the sensor checks the temperature periodically and stores it into an internal register. The Reader can request the actual value in the register anytime. This value represents the highest measured temperature during the polling period. One application scenario could be the detection of fever or the measurement of the body temperature generally. Inputs and outputs are shown in Figure 3.22.

- **SENSOR_COMPARE**

SENSOR_COMPARE is directly connected to the internal ADC. This ADC measures the temperature and informs this unit, when the conversion is finshed. It is enabled, if en_polling_i is set. The ADC reads out the actual temperature and informs the unit via adc_rdy_i. Then SENSOR_COMPARE has to store the value into a 10 bit register, if the input value is higher than the already stored value.

- **TEMP_MEAS**

All data, clocks and enable signals are provided by the master digital part, of which the Bus Arbiter is part of. If en_polling_i changes to high, it is stored in a register along with 2 configuration bits. This is necessary, because the configuration bits and the signal will not be available during the whole polling period. The configuration bits determine the polling interval. The polling counter is clocked with 1 kHz. The interface to the ADC is clocked with 1 MHz.

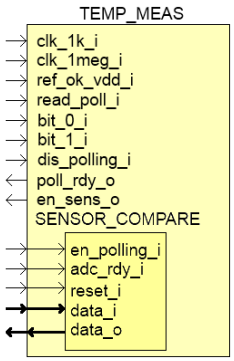

Figure 3.22: Ports of the Polling unit

After the Reader has transmitted the command and set the polling time with the 2 configuration bits, the internal clock counter is activated. It counts until it reaches the desired value, which depends on the configuration bits. Table 3.1 is relating the bits and the corresponding polling interval.

Bit pattern	Interval
00	100 ms
01	1 s
10	1 min
11	10 min

Table 3.1: Polling configuration

Figure 3.23 illustrates the function of this unit after the counter has reached its final value. The counter restarts and en_sens_o is set. This ensures that the ADC is activated. After approx. 500 μs the ADC has finished and the counter with the clock of 1 MHz is activated. After approx. 10 μs en_sens_o can be reset and the value is stored in the internal register of the Polling unit.

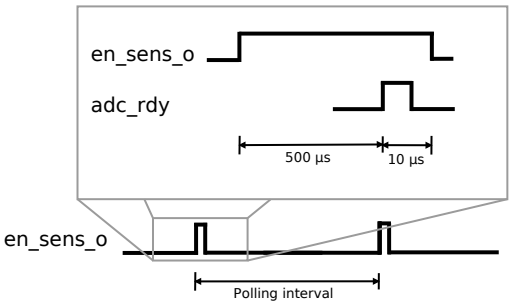

Figure 3.23: Desired functionality of the Polling unit

Figure 3.24 shows the life cycle of the polling mode. If the Tag is within the field, the Reader supplies the main digital part with energy and transmits the configuration data along with the activation sequence. The Polling unit takes over the data and writes down the 2 bit configuration. Before the process is started, the main digital part ensures, that the battery is able to supply the polling component with energy. The oscillator for the clock counter is started and now the Polling unit drains its energy from the battery. The Polling process is started until the Reader requests the data. Therefore it deactivates the Polling process with setting dis_polling to high and reads out the data register. The Polling sequence can then be started again.

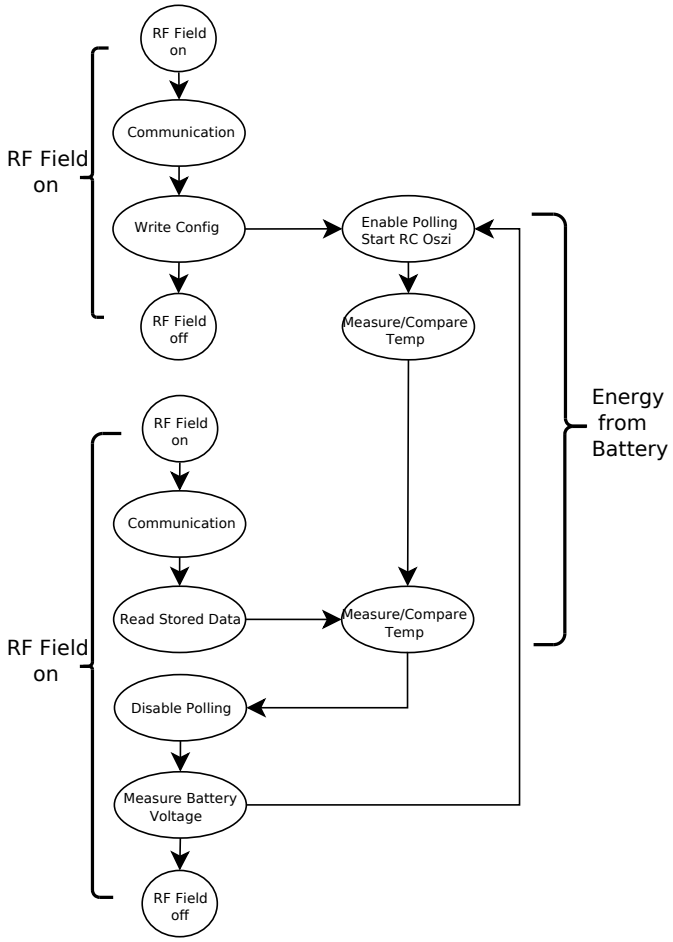

Figure 3.24: Polling sequence

Chapter 4: Simulation Results

The system components are coded in VHDL. The RTL Description of the components is coded to be synthesisable. Additionally an existing testbench was used to verify the correct functionality of the system. The simulation environment consists of the Software *QuestaSim 6.6d* from Mentor Graphics [6].

4.1 Introduction

The NFC functionality was implemented and simulation results are presented in this chapter. The Simulation Setup is illustrated in Figure 4.1. The main part of it is the Design Under Test (DUT). It consists of the Multimaster Multislave Bus System. Behavioural models represent the expected environment of the digital system of the chip. This environment includes the AFE which forwards the Readers commands along with the reset and clock signal to the RTL Description. The RISCs ROM is connected via a program counter port (pc) and an instruction port (inst). Its programming is defined via an input file. Also the EEPROM or NVM and its signals are connected with the DUT. The inital values of the NVM are defined in a file.

The off-chip environment is represented by the EPC Reader and the NFC Reader. These 2 models emulate the RF communication signals which are received by the AFEs. The command sequence is defined via an input file.

This chapter will focus on the simulation of the NPU as part of the DUT and the Polling component. Tests include proper anticollision processing, memory operations and sensor activations. The correct operation of the CTS was verified in hardware already.

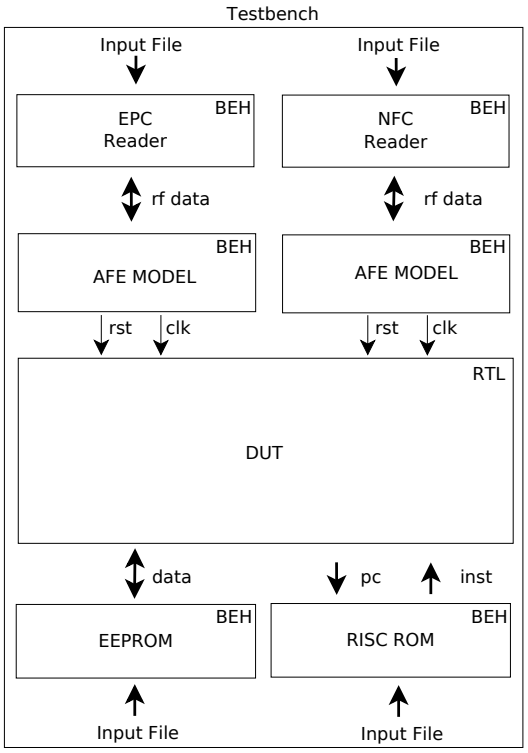

Figure 4.1: Simulation Setup

4.2 Decoding

This section is explaining the decoding mechanism for input signals as described in Section 2.3.1. Figure 4.2 shows the decoding of a REQA command. Therefore pad_la_i represents the encoded input from the PCD. clk_s is the 848 kHz clock. If new data is available, CLUART stores every received bit into the register data_s. While it is receiving a byte, the RISC is halted via cluart_halt_s. The bitstream is delivered by the AFE. The modulation within a bit duration of approx. 9.43 μs determines the received bit. This duration is equivalent to 8 periods of the subcarrier with a frequency of 848 kHz. In detail the CLUART is a state machine with states depending on the received data. The simulation shows that 7 bits are loaded into the data_s register, so that one bit remains undefined. 0x17 signals the NPU how much bits are valid. This is labelled with a circle in the simulation figure. With this information the NPU can deal with this problem. By using the first 7 bits only it encodes the REQA command (0x26).

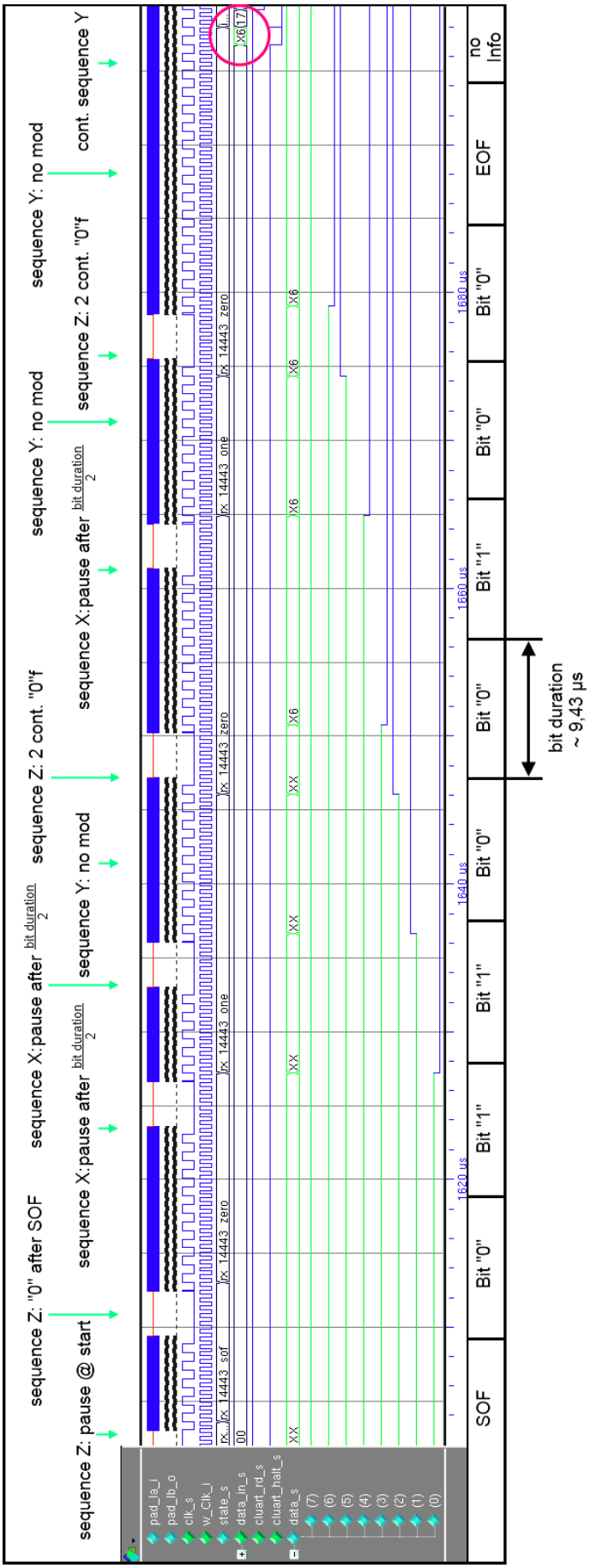

Figure 4.2: A decoding example

4.3 Load modulation

As described in Section 2.1 the signal from PICC to PCD is transmitted via the load modulation mechanism. The subcarrier modulation is represented in Figure 4.3 through pad_lb_o. The modulation runs with 848 kHz. As a response to a REQA command the PICC transmits 0x0050, known as ATQA. For demonstration purposes the transmission of the first byte is shown in Figure 4.3. Before the transmission can start the RISC has to deliver the number of valid bits to CLUART. After that the RISC has to apply the data byte to the bus. Therefore the RISC sets wr_i to "1" and waits until CLUART is ready to accept new data. If CLUART is ready the data in the register data_s is adopted and CLUART can start the transmission. It has to wait until the FDT passes before it can transmit data.

After that it starts with a Start of Frame followed by a data byte, an odd parity bit and End of Frame. The modulation corresponds to the scheme introduced in Section 2.3.1. The port pad_lb_o is connected to the AFE and ensures that the modulated signal is transmitted via the antenna. Immediatly after delivering the first byte the RISC tries to deliver the second byte to CLUART by holding wr_i at "1". It can be seen that CLUART suspends the RISC until the transmission is finished. When it is ready to transmit the second byte it resets cluart_halt_s. It accepts the new data sets cluart_halt_s to "1" and starts a new transmission procedure to forward 0x00. This is not illustrated in Figure 4.3.

It is crucial, that the RISC is fast enough at applying new data to the bus and at adopting received data. The CLUART is bound to certain reception and transmission timings. In the worst case it is possible that a data byte is lost, because the RISC is not fast enough. Proper timing was ensured by supplying the whole system with a clock of 848 kHz. Thus the RISCs operation speed was fast enough to meet the timing requirements of the CLUART unit.

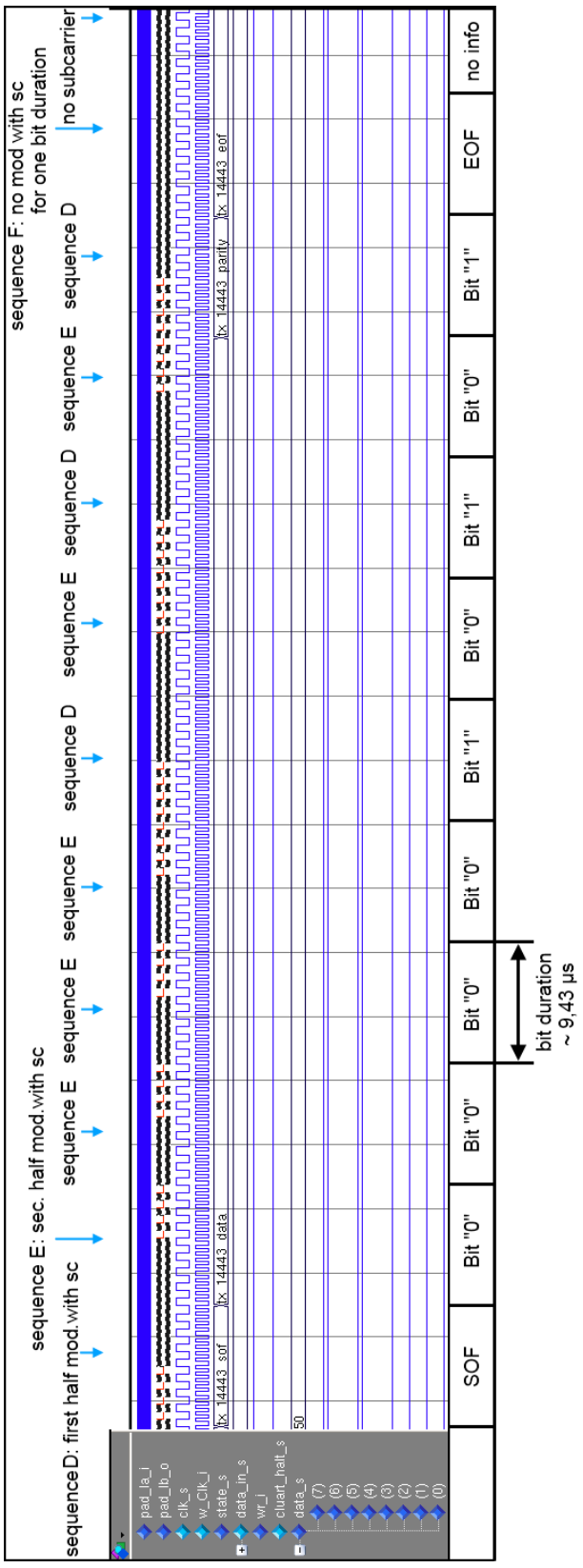

Figure 4.3: Response to a REQA command

4.4 Protocol and chip functionality

The protocl processing should work as described in the NFC standard in Section 2.5. For demonstration purposes the reception of a Select command and the proper response are illustrated and discussed. Furthermore the simulation results for the Polling component are reviewed and compared to the expected behaviour.

4.4.1 Reception of a Select command

The reception of the first select bytes SEL (0x93) and NVB (0x20) is shown in Figure 4.4. The RISC requests data with setting rd_o to high and applying the address of CLUART to the bus. A received data byte is stored in the register rfs_0. The information (valid bits, EOF) about the received byte is stored in rfs_1. Of course the same storing procedure is performed by receiving all other data like REQA or WUPA. During reception halt_o is set. When the first byte is received by CLUART it releases the RISC by resetting halt_o and providing the data and additional information about it. Afterwards the reception of the next byte can continue.

EOF is indicated by the 4^{th} bit of the register rfs_1. As long as this bit is "0" the RISC requests new data. In case of a full byte, the last received byte will be received twice. In Figure 4.4 this happens during the reception of the byte 0x20 so that EOF is noticed at the next readout from CLUART.

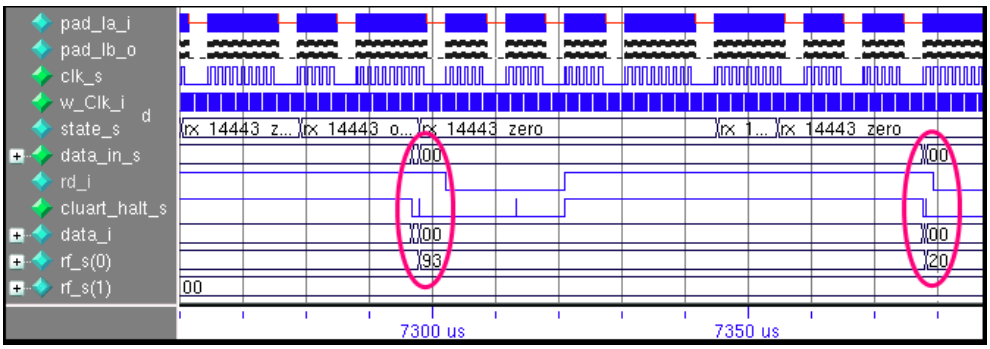

Figure 4.4: Reception of a Select command

4.4.2 Transmission of a response to a Select command

In chapter 3 the used memory structure was introduced. The memory structure contains the UID which is used for the prototype. The first level UID consits of 4 hexadecimal values. These are 0x02, 0x03, 0x04 and 0x8D as BCC. The second level UID counts from 0x05 to 0x08 with a BCC of 0x0C.

A response to the first select command is shown in Figure 4.5. The first transmitted byte in the first anticollision level is 0x88 and is provided by the RISC. The CLUART takes over the byte and transmits it bitwise. The byte with the value 0x88 is called cascade tag and signals the PCD that the UID consists of 7 bytes. CLUART is not busy, so 0x88 is transmitted immediately, after being applied to the bus by the RISC. The transmission of the other bytes will occur when the transmission of the previous byte was finished. The register data_in_s of CLUART holds then the data which has to be transmitted and is valid when both data and status byte were delivered to CLUART. Otherwise it could lead to errors, if the CLUART timing is not met. The last read procedure is started by the RISC in order to continue with the anticollision procedure as described in Section 2.4. Meanwhile the CLUART transmits an odd parity bit and the EOF.

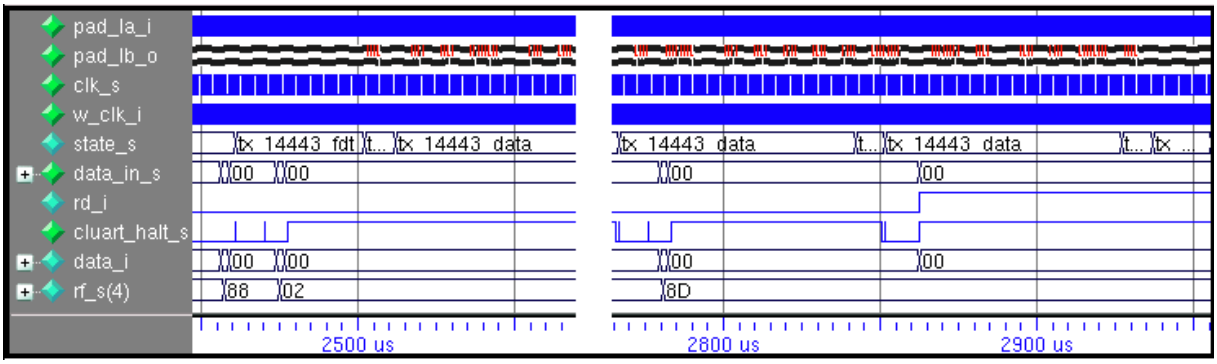

Figure 4.5: Response to a Select command

4.4.3 Temperature measurement

After selecting a specific PICC, Temperature measurement can be started. Now the chip is in the ACTIVE state and it accepts READ and WRITE commands. Measurement is started by writing a specific pattern to the sensor configuration register, which is illustrated in Figure 3.8.

Since the chip has 2 sensors integrated, they are activated by setting 2 different bits to "1" within the sensor configuration word.
The activation sequence for the first sensor is illustrated in Figure 4.6. As described in Figure 3.21 the state machine starts in the state GET_COMMAND. After receiving 0x90 (circle in Figure 4.6) the address is stored into a temporary register. It consits of the 5 LSBs of nteg_ctrl_sfr_in. In this special case the blocknumber is 0x10000 and is mapped to 0x76 and 0x77 according to Section 3.4.6. The MSB determines whether data is written or read. A set MSB leads to a write procedure.

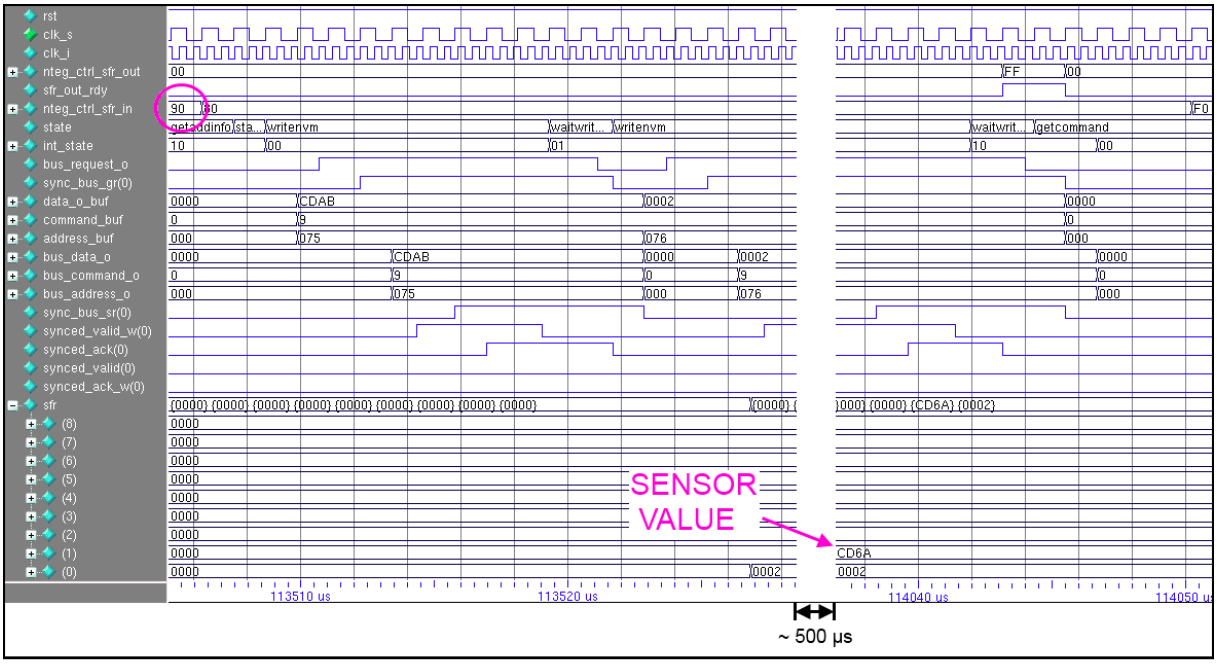

Figure 4.6: Temperature Sensor Activation

The bridge has to deal with two clocks, because CTS, SFRs and NVM are supplied with another clock frequency than the NPU. Therefore clock synchronization is required.

Also this protocol writes exactly four bytes each time. Therefore the address has to be read out with a READ command before starting a write cycle to ensure that no data is overwritten. Anyway here the stored data (0x0000) is overwritten with 0xABCD to demonstrate the functionality of the bus. After applying a synchronized bus request to the Bus Arbiter and preparing the data, command and address the state machine waits for bus clearance. This is signaled with the input bus_granted_i and its synchronized version sync_bus_gr. In WRITE_NVM data, command and address are applied to the bus and it is waited for an acknowledgement, that data has been adopted. WAIT_WRITE_READY waits for a bus_slave_rdy to continue the write cycle. Since Blocknumber 0x10 references to addresses 0x76 and 0x77, the first word was written to a General Purpose SFR. The second word works as sensor configuration and corresponds to the first register of the sensor SFR. It activates the temperature sensor with 0x0002.

The bridge waits approximately 500 μs before the ADC has finished operation. The new temperature is stored in the second register of the sensor SFR. Also the bridge signals the RISC with 0xFF that data has been successfully written. After receiving a ready signal the bridge gets a finishing byte (0xF0) and CLUART transmits an acknowledge to the Reader to signal a succesful WRITE command.

The same process as explained above is performed during shunt sensor activation. The difference is the smaller processing time of the ADC of approximately 100 μs. The procedure of reading out the stored sensor value is illustrated in Figure 4.7. It consists of eight read requests for 16 bytes of data. A read request consists of the READ command, and the address. The Bridge ensures data integrity through synchronization. The first data word is a shunt sensor value, which is applied to the bus through nteg_data_sfr_out. The remaining sensor values are 0x00. Shortly after the read procedure the gathered data is transmitted.

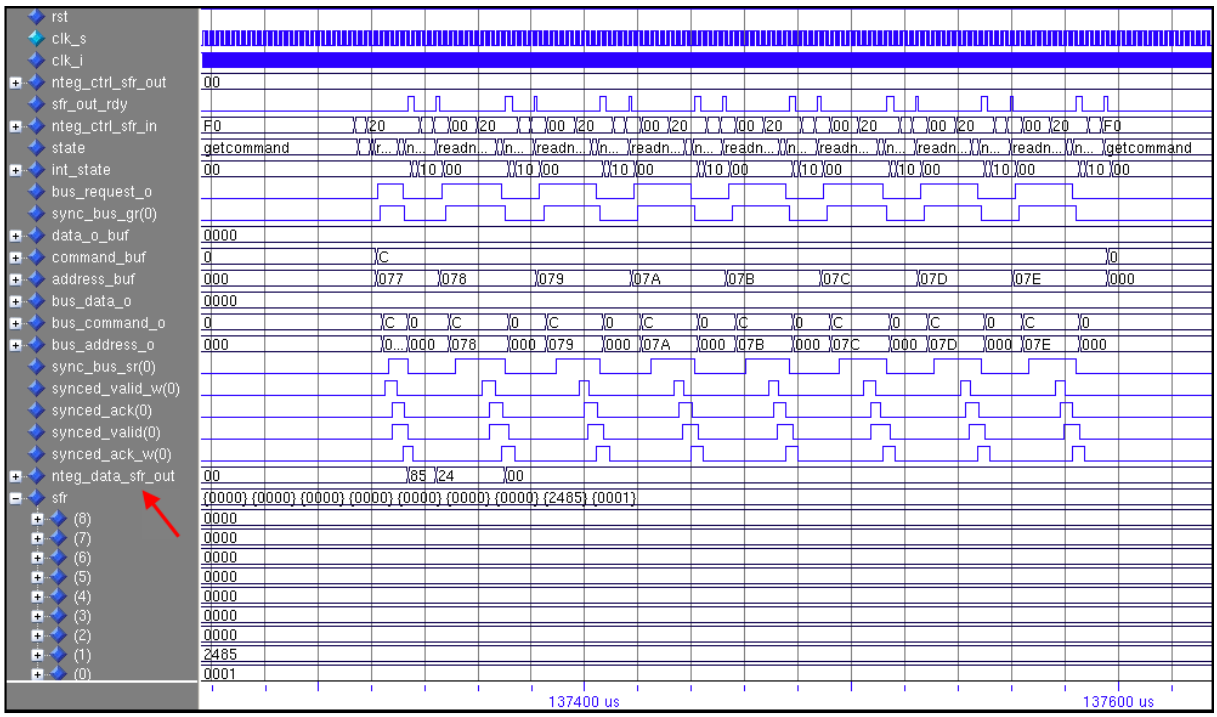

Figure 4.7: The value of the shunt sensor is read out

4.4.4 Temperature Polling

As described in Section 3.5.3, this unit measures the temperature and delivers the highest value. Figure 4.8 shows two different configurations. It first measures temperature every 100 ms, because both config bits are “0”. When the bit configuration changes to “01”, the polling cycle is 1 s. If no energy is available, the last value is retained.

Figure 4.8 features also a closer look on the activation process of the ADC. The functioning corresponds to the explainations in Section 3.5.3. During the polling period the unit is supplied with a clock of 1 kHz. If en_sens_o changes its state to high the ADC is activated in the background and requires approximately 500 μs until it sets adc_ready to high. The unit is then supplied with a 1 MHz clock in order to reset en_sens_o after approx. 10 μs. The data applied to port dat_o represents the last measured value.

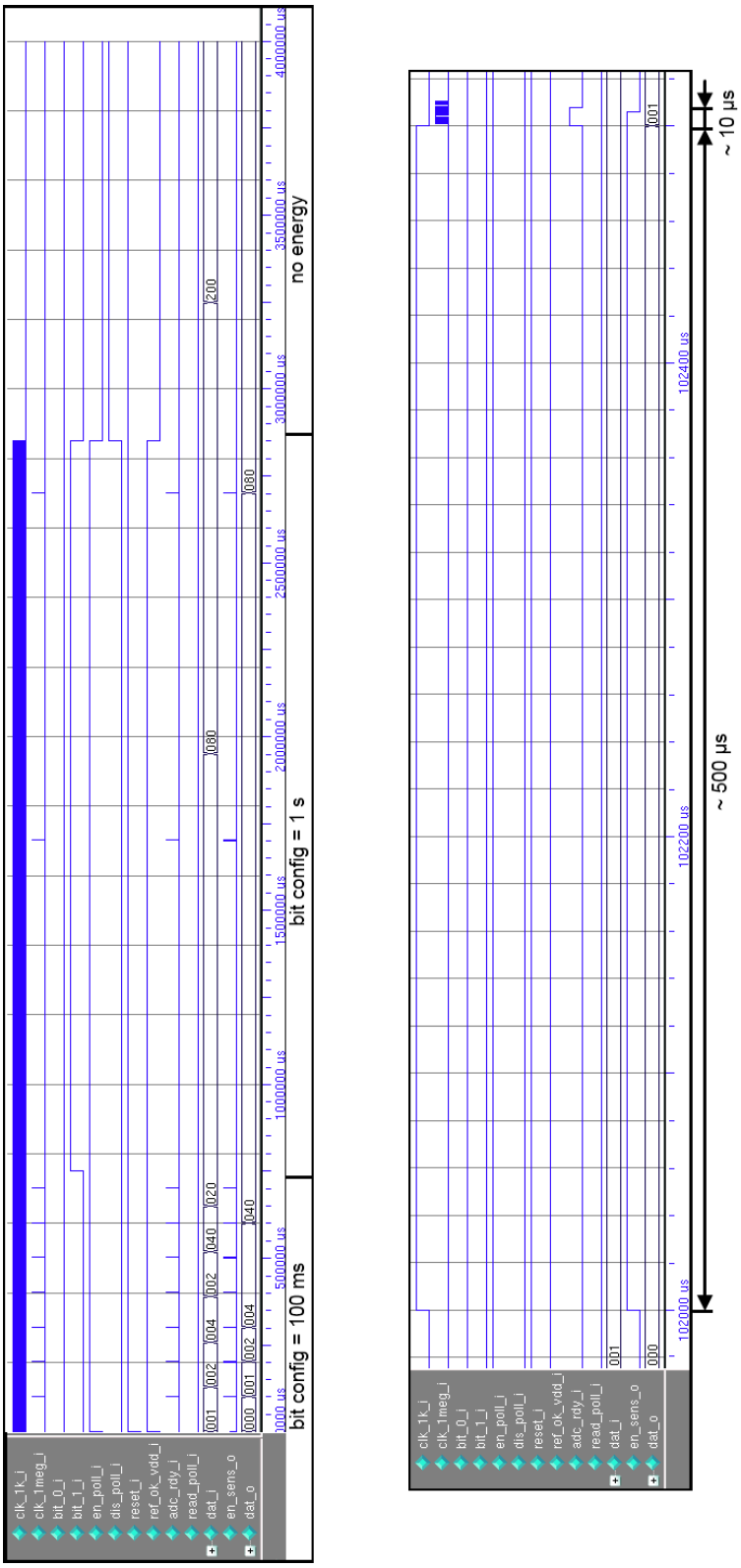

Figure 4.8: Read out of shunt sensor data

Chapter 5: Synthesis Results

The synthesis was performed for the designed subblock of the system called NPU, to test its functionality in a gate level simulation. Also the Polling unit was synthesized and verified. The synthesis is based on a 130 nm CMOS fabrication technology and is performed using the *Synopsis Design Compiler* [15]. The gate level simulations were finished for both parts sucessfully and correspond to the results of the Chapter before.

The area in Table 5.1 and Table 5.2 is given in μm^2.

5.1 NPU Area Report

The Area Report is consisting of the RISC, BRIDGE, CRC, CLUART and RAM. The ROM will be added externally.

Combinational area	24288.480567
Noncombinational area	36226.079409
Total cell area	60514.559976

Table 5.1: Area Report for NPU

With a dimension of 6,71 μm^2 per NAND2 gate in the used process the equivalent gate count of the combinational area is 3620 gates. The cell area of a posedge triggered D Flip-Flop with an asynchronous reset covers 38.8 μm^2. This means that the sequential area consists of 934 Flip-flops. The area of the ROM amounts 16000 μm^2. The whole unit covers an area of 76514.559976 μm^2.

5.2 Polling unit Area Report

Combinational area	1601.600015
Noncombinational area	1734.399971
Total cell area	3335.999986

Table 5.2: Area Report for the Polling unit

The combinational area is made up of 240 gates and the sequential area contains 45 Flip Flops.

Chapter 6: Measurement

Until now only the Polling unit was produced and verified along with the previous version of the CTS. After production the chip was tested under laboratory conditions. Therefore communcation was established and the functionality was tested.

6.0.1 Polling cycle

When Polling is activated, the chip drains the energy from a 3 V battery. The current consumption in IDLE mode is 1-2 nA, while it needs 4 μA during ADC operation. This consumption has been optimized for the next tapeout in order to reduce power consumption. It can be seen in Figure 6.1 that the ADC is active 300 μs after it is regulated to 1.5 V. This voltage is necessary to produce a sufficiently precise output. $V_{dd\_polling}$ delivers the power to run the polling clock.

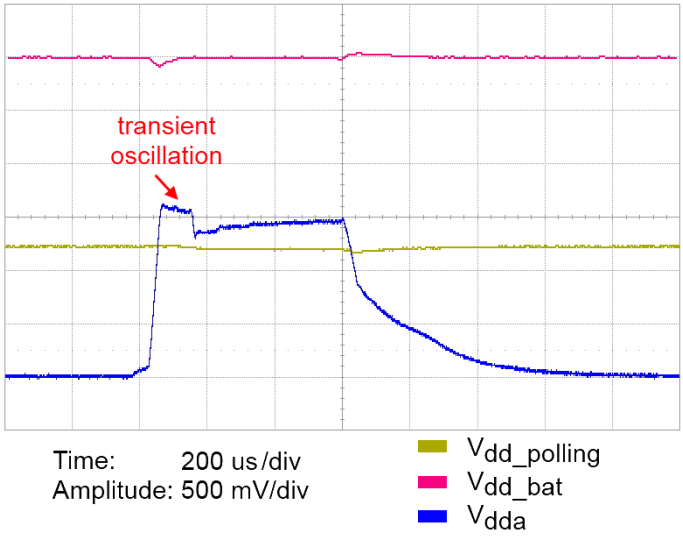

Figure 6.1: Active ADC during polling mode

Figure 6.2 represents another version of Figure 6.1 showing the polling cycle. The configuration bits were set to "00" to achieve a polling cycle of 100 ms. Due to the fact that the expected polling frequency differs from 1 kHz, the polling cycle exceeds 100 ms but appears regularly. The data can be read out and represents the current environment temperature.

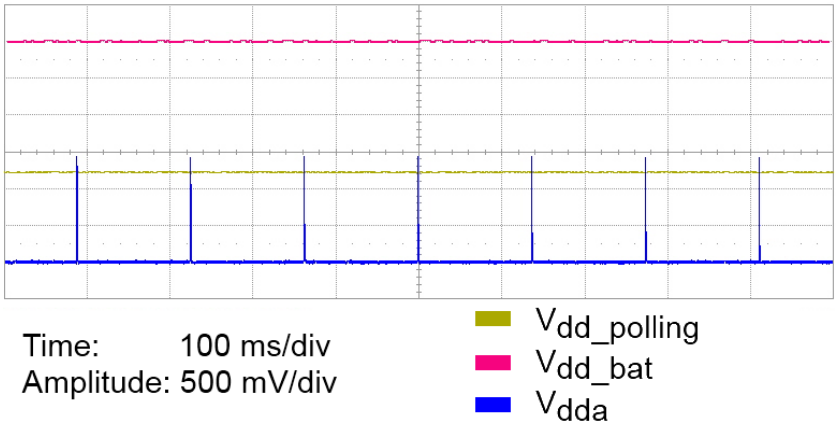

Figure 6.2: Polling cycle

Chapter 7: Conclusion

This chapter summarizes the topics covered by this thesis. Furthermore the results and further work is discussed.

7.1 Summary and Results

The aim of this thesis was to design and verify a digital Communication Front End for NFC support. Due to in-house delays the CLUART and RISCs functionality has to be emulated in a testbench, until it was deliverd. This allowed the design of the other digital units. Furthermore they were attached to the existing system with Bus Master, SPI, NVM and sensors.

First, the application scenarios for the chip were discussed. Afterwards the used protocols were presented and explained. Additionally some important considerations in digital design were introduced. These are important to ensure correct functionality of a digital system. Then the concept for the digital core was presented along with the bus system, sub units and their interconnection. Furthermore the new Communication Front End was introduced and its RTL implementation was explained. Aftwerwards the RTL Description was simulated and the results were presented. The synthesis of the NPU and the Polling unit is also finished. The results of the synthesis and measurements were shown and discussed.

The NPU meets the requirements of the ISO 14443 or NFC protocol related to timing and communication sequence. Also the synthesized versions of the NPU and Polling component were verified and tested with the testbench. Due to the memory area of the testchip has to be kept unlocked, the locking mechanism was not implemented.

The advantage of using the RISC architecture is higher flexibility at programming due a standardized interface. Assembler as programming language is easy to learn and leaves room for optimization of speed, area and power consumption. Additionally post silicon bug fixing is eased, because the last metal mask can be changed as last step of the production process. The RISC and its ROM are also equipped with a clock gating

mechanism. In order to reduce power consumption both are not supplied with a clock, if the RISC is halted.

7.2 Further Work

Later when other parts of the system are improved and finished, the whole system can be synthesized. Furthermore UPF (Unified Power Format) will be implemented in order to minimize power consumption, because CTS and NPU are not supposed to work simultaneously. Therefore one of the Communication Front Ends will be shut down if not used. At the end the chip will support EPCglobal UHF and NFC in HF. Additionally it will be researched whether the whole chip can operate with 848 kHz (during NFC), because the write and read performance of the NVM depends on the frequency used. If this can be achieved without errors, no synchronization will be necessary in the bridge during WRITE and READ commands. Furthermore another student will continue the work with the used RISC. In future the RISC should be responsible for the processing of both protocols.

Bibliography

[1] MAS project. online, March 2012. http://www.mas-aal.eu/.

[2] OpenPCD Passive RFID Project. online, March 2012. http://www.openpcd.org/ISO14443/.

[3] Andrejka, T.: Implementierung der EPC Class-1 Generation-2 RFID fuer ein Comprehensive Transponder System. Hagenberg School of Informatics, Communications and Media, 2007.

[4] Finkenzeller, K.: RFID HANDBOOK. John Wiley & Sons Ltd, 2003.

[5] Forum, NFC: ISO/IEC FDIS 14443-3:2000(E): Type 2 Tag Operation Technical Specification. online, July 2007. http://www.nfc-forum.org/specs/spec_list/.

[6] Graphics, Mentor: Modelsim. online, 2012. http://www.mentor.com/products/fv/modelsim/.

[7] Grout, A. I.: Integrated Circuit Test Engineering: Modern Techniques. Springer Verlag London Limited, 2006.

[8] Herndl, Thomas: Presentation for the MAS Project for CRE. internal, 2011. internal presentation.

[9] Inc., EPCglobal: EPCglobal Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz Version 1.0.9. online, 2004. http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.

[10] Kaeslin, H.: Digital Integrated Circuit Design. Cambridge University Press, 2008.

[11] Langer, J. and M Roland: Anwendungen und Technik von Near Field Communication(Nfc). Springer Heidelberg Dordrecht London New York, 2010.

[12] Nokia: Understanding NFC Data Exchange Format (NDEF) messages. online, 2012. http://www.developer.nokia.com/Community/Wiki/Inside_NFC:_Understanding_NDEF_message.

[13] Standardization, Geneva, Switzerland. International Organization for: ISO/IEC FDIS 14443-2:2000(E): Part 2: Radio frequency interface power and signal interface.

online, July 2000. http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=50941&ICS1=35&ICS2=240&ICS3=15.

[14] Standardization, Geneva, Switzerland International Organization for: ISO/IEC FDIS 14443-3:2000(E): Part 3: Initialization and anticollision. online, July 2000. http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=50942&ICS1=35&ICS2=240&ICS3=15.

[15] Synopsis: Synopsis Design Compiler. online, 2012. http://www.synopsys.com/home.aspx.

[16] Tinder, F., R.: Engineering Digital Design. Academic Press, 2000.