UiO : Department of Physics
University of Oslo

Single ASIC NFC Sensor Tag

Joar Digernes Særsten
Master's Thesis, Spring 2016

Abstract

In order to use a smart phone or smart watch with the near field communication (NFC) standard to read sensor data from a micro-implant, an NFC tag with integrated ADC and sensor front-end is implemented.

An NFC digital protocol module is combined with an energy efficient front-end, ADC, energy harvester and NFC data link to form a passive NFC tag with a fully integrated and complete sensor interface.

This work builds on previous work where parts of this system have been designed and tested as standalone solutions. Previously, a sensor interface ASIC has been developed that provides the front-end to a resistive or capacitive sensor and converts its output (e.g. blood sugar concentration) into digital data ready to be transferred. Another NFC physical layer ASIC has been developed which hosts the energy harvesting and physical communication layer required to operate with an NFC enabled mobile phone, or other NFC readers. An implementation of the NFC digital protocol has been created and tested on a FPGA together with the physical communication layer and front-end and ADC. The FPGA implementation was modified further to be synthesized and combined with the two previous ASICs into a single ASIC.

A power-on reset circuit was implemented for the digital core's startup conditioning on power up.

The main contributions of this work has been to add the higher level NFC protocol control logic module as well as a power-on reset circuit, and to integrate all of these modules onto a single system-on-chip (SoC).

Acknowledgment

First of all, I would like to thank my advisors, Dr. Ali Zaher and Prof. Philipp Häfliger, for giving me the opportunity to write my master thesis on such an interesting subject. I would also like to thank them for their advice, guidance and willingness to answer both the hard and the surprisingly simple questions that I have encountered during the process of completing this master project.

Thanks are in order to Girish Aramanekoppa Subbarao who spent much time cutting the ESD protection connections using the Focused Ion Beam machine.

I would also like to thank Olav Stanly Kyrvestad, Senior Engineer here at the Research Group for Nano electronic systems, who has been of substantial help throughout the project.

And to my fellow students at SEF, ELDAT and NANO: thanks.

Oslo, May 2016
Joar Digernes Særsten

Contents

List of Figures

List of Tables

Preface

This thesis is the result of my study for a master degree in Microelectronics at the Department of Physics, University of Oslo. I started this work spring 2015 and concluded my research spring 2016.

My interest into biological sensing implantable technology and embedded systems were the main reasons for choosing this project. A personal curiosity was to be a part of the complete process from design concept to a complete system which could interface with the real world. This specific project looked to not only be an interesting one, but one with potential future real life implementations.

Ali Zaher (my co-supervisor), Thanh Trung Nguyen, Philipp Häfliger (my main supervisor), and I published a paper called “the Integrated electronic system for implantable sensory NFC tag”, seen in [22]. It is a preceding and concurrent part of this work, where the parts implemented on a single ASIC in this work are presented as separate parts of a system with a goal of a future single ASIC implementation of said system.

Part I

Introduction

Chapter 1

Goals for this thesis

Goal: Demonstrating how a small implantable NFC and sensor tag is feasible for chronic disease monitoring, e.g. diabetes or high blood pressure.

Objective: Design, implement and test a single passive ASIC NFC sensor tag to enable a small NFC sensing system.

Scope:

- Make and test a process for digital design synthesis.
- Synthesize an NFC-A Tag type 1 digital core.
- Implement previously made sensor readout, and NFC communication and power layer modules with synthesized digital core in a single ASIC.
- Create a system which only needs an external antenna, capacitive energy storage and sensor.
- Create system testing board.
- Connect test board together with an FPGA development board and do comprehensive testing.
- Test system with commercial, commonly available, NFC reader: smartphone with NFC capability.
- Demonstrate passive tag operation, that is, no batteries or external power needed on the tag's side
- Demonstrate tag sensor read-out using an NFC reader.

Chapter 2

Background

2.1 Motivation

In 2006 Philipp Häfliger here at the Research Group for Nano electronic systems at the department of informatics at the University of Oslo, started as a collaboration with a medical technology company, Lifecare AS, to research a continuous and long term glucose sensor read-out. This project has been called GlucoSense.

There are an estimated 415 million people with diabetes per 2015 in the world. With conservative estimates, the global prevalence of diabetes type 1 is 29 million (7% – 12% of diabetes cases in high-income countries are type 1). With daily insulin treatment, regular blood glucose monitoring and maintenance of a healthy diet and lifestyle, people with type 1 diabetes can lead a normal, healthy life.[6]

An implantable glucose sensor together with the NFC sensor read-out tag presented in this work could reduce the need of daily blood-drop glucose measurements or weekly replacement of a percutaneous sensor.

2.2 Previous work

A first integrated sensor tag electronics implementation is presented in [10]. It started as a collaboration between the Research Group for Nano electronic systems, here, and Lifecare AS, a health startup.

Thanh Trung Nguyen together with Philipp Häfliger developed front end and ADC further into what is presented in [18] and [13]. [13] and [10] also uses only a single resistance divider instead of a Wheatstone bridge, while maintaining a differential input by swapping the polarity of the sensor supply source which suppresses 1/f noise since it uses correlated double sampling (CDS). The RF back-end is described in the extending paper [15]. There, a parallel to serial logic block continuously encodes read-out data as a serial bit stream, which in turn is transmit back to the reader using load-shift keying (LSK). In [12] the ADC as is implemented here is demonstrated, that is, it envisions a capacitive sensor to be measured, similarly to this project. [16] demonstrates [12] implemented in 90nm TSMC process; its layout was refitted for use in the project presented here.

Ali Zaher, Aage Dahl and Thomas Peter Plagemann developed an android NFC performance testing software suite in [5]. Based on their measurement results from testing five different NFC tags, NFC-A tag type 1 was selected for further development based on its low power consumption during read and simplicity. It did have higher power consumption during write operations, but as *write* is not implemented with actual writing of bits, in this project, it should not affect the technology selection much.

In [22] Ali Zaher, Joar Særsten, Thanh Trung Nguyen and Philipp Hafliger, presents the system implemented with the three main components being two ASICs and one FPGA, which again is implemented on a single ASIC in the project presented here.

2.3 NFC-A tag type 1

NFC-A and its subset tag type 1, is the communication protocol used in this project.

NFC-A has two communication schemes; Reader to tag communication, poll → listen, uses amplitude shift keying(ASK). And Tag to reader communication, listen → poll, which modulates the load on the receiving coil, also known as load-shift keying (LSK). [8]

2.3.1 Tag to reader communication

Communication from tag to a reader is accomplished with subcarrier modulation using Manchester encoding.

The tag creates the sub carrier by switching a load connected to the inductive coupling area, creating a subcarrier frequency of $\frac{fc}{16} \approx 847\text{kHz}$ as per [3]. This subcarrier is subsequently keyed on and off according to Manchester encoding.

In Manchester encoding a binary 1 is represented by a negative transition in the half-bit period and a binary 0 is represented by a positive transition. [7, page 181] Tag to reader bit representation and Manchester encoding definitions are listed in Table 2.1 where a bit duration is $\frac{fc}{128} \approx 106\text{kHz}$.

Table 2.1: Tag to reader bit encoding, from [3]. Subcarrier is $\frac{f_c}{16} \approx 847\text{kHz}$

Logic value	Subcarrier modulation of carrier
1	First half modulated
0	Second half modulated
Start of communication	First half modulated
End of communication	no modulation

The frame format for tag to reader communication is: a start bit, data byte sent with least significant bit first, followed by an odd parity bit, repeated until an *end of frame* pattern is transmitted to signal the end of tag to reader communication. Figure 2.1 illustrates the tag to reader frame format.

The *end of frame* pattern is straightforwardly defined as no modulation. [17]

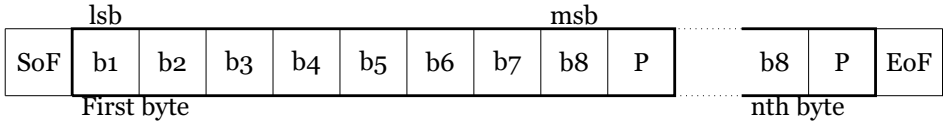


Figure 2.1: Tag to reader frame format. Eof is End of frame, SoF is Start of Frame.

2.3.2 Reader to tag communication

Communication from the reader to the tag is accomplished by modified miller encoding using amplitude modulation keying (ASK) of the carrier wave [8, page 15].

In normal Miller code (also known as delay encoding): A binary 1 is represented by a transition in the half-bit period; a binary 0 is represented by the continuance of the 1 level over the next bit period. A sequence of zeros creates a transition at the start of a bit period, so that the bit pulse can be more easily reconstructed in the receiver (if necessary). [7, page 181]

The encoding used here, for reader to tag communication, is a modified miller encoding scheme, where each transition is replaced by a negative pulse.

The pulses are shorter than a bit period ensures that the carrier wave is active for longer and therefore more power is transmitted during communication [7, page 181].

During ASK modulation the reader pulls the carrier amplitude down to less than 5% of its initial amplitude. These pulses lasts for a minimum of 0.5µs or up to a maximum of 3µs.[3]

A more rigorous way of defining this modified miller encoding, as listed in [3], is to define each transition pulse type into patterns, and listing how bit values and their order determine what patterns should be expressed. There are three Poll→Listen patterns, **X**, **Z** and **Y**, they are generated based on the rules listed in Table 2.2.

Table 2.2: Poll→Listen pattern modulation, as per [8]. f_c is the carrier frequency, typically 13.56MHz

	Bit duration($\frac{128}{f_c}$)	
Pattern	First half	Second half
X	No modulation	Modulated
Y	No mod.	No mod.
Z	Modulated	No mod.

Bits are encoded into patterns using the rules listed in Table 2.3. Shortly summarized, 1s are encoded as X patterns, while 0s are encoded as Y and

Z patterns, where Z patterns are used for repeated zeros, Y for first and singular 0s. The take home message from this table should be that a Y pattern is never repeated, and only follows an X. The only instance when a Y follows a Z is after a transmission has ended, that is, no communication. The Y pattern is not really a pattern, as it involves no modulation of the carrier wave; which is also why Y pattern can indicate *end of frame* (EoF) as no modulation for subsequent bit durations means no frame is transmitted.

To indicate the start of frame (SoF), a Z pattern is expressed, this gives the tag something to coordinate its timing to, as it is always a Z pattern.

The tag (Listen device) decodes patterns according to the rules listed in Table 2.4.

Table 2.3: Poll → Listen bit encoding, based on previous bit value. Start of Frame (SoF) indicates to a tag that a frame follows.

Previous bit	current bit	pattern
None	SoF	Z
1|0	1	X
0	0	Z
1	0	Y

Table 2.4: Poll→Listen pattern decoding. Start of Frame (SoF) indicates to a tag that a frame follows.

Pattern	Interpretation
First Z	SoF
X	1
Z	0
Y after X	0
Y else	EoF

The frame format for reader to tag communication is what is specified in [17] and [8] under Tag type 1. NFC-A, and tag type 1, frame format uses 7-bit short frame for commands, which starts with a *start of frame* (SoF) pattern and ends with a *end of frame* (EoF) pattern. Bits are transmitted in the order of least significant bit (lsb) to most significant bit (msb).

NFC-A tag type 1 is different from NFC-A specification in that it breaks up bytes into single frames instead of sending them together as a continuous bit stream. Regular data is transmitted similarly, but instead of seven bits, all eight are transmitted. Figure 2.2 describes these two frame formats.

2.3.3 Commands

Table 2.5 lists the reader commands as defined in [17] and the corresponding overlapping commands from [8].

Short or Command Frame

SoF	b1	b2	b3	b4	b5	b6	b7	EoF
	lsb						msb	

Operand or Data Frame

SoF	b1	b2	b3	b4	b5	b6	b7	b8	EoF
	lsb							msb	

Figure 2.2: Reader to tag frame format

Table 2.5: Reader(poll) to tag(Listen) commands[8, 17]

NFC-A	Topaz/Tag type 1	Command code	Description
SENS_REQ	REQA	$0x26$	Request command
ALL_REQ	WUPA	$0x52$	Wake-up
	RID	$0x78$	Read ID
	RALL	$0x00$	Read All
	READ	$0x01$	Read(Single byte)
	WRITE-E	$0x53$	Write-with-erase
	WRITE-NE	$0x1A$	Write-no-erase

REQA and *WUPA* commands, should, from the specifications, respond with an ATQA, defined as two bytes without any checksum. The first byte, $0x00$, means that the tag UID is 4 bytes long. The second byte is $0x0C$, and means that the tag is of type 1 tag platform. [8] This is again explained in Table 2.6.

First Byte:

b8	b7	b6	b5	b4	b3	b2	b1
0	0	0	0	0	0	0	0
ID size 4 bytes		RFU	Single device detection type 1 tag				

Second Byte:

b8	b7	b6	b5	b4	b3	b2	b1
0	0	0	0	1	1	0	0
RFU				Type 1 Tag			

Table 2.6: ATAQ response

Figure 2.3 demonstrated how a Wake-up (*WUPA*) command is encoded for reader to tag (poll → listen) communication.

In this project the *WRITE-E* command is also used to enable the ADC clock. It was selected for that purpose because of its long response time at about 5.23ms. [17] Which means that the reader supplies uninterrupted

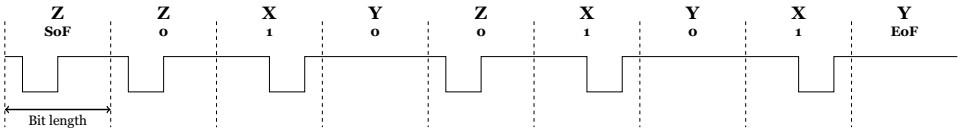


Figure 2.3: WUPA poll modulation waveform

power and clock for the whole duration, and the ADC can work without interruption during that time.

2.3.4 Waking a tag

A NFC-A tag, and NFC-A Tag Type 1, are woken up with a *WUPA* or *REQA* command before actual reading or writing can be done. The tag must be ready to receive one of these commands within a guard time of 5ms of unmodulated carrier wave. [8]

2.3.5 Cyclic Redundancy Check

To ensure that commands and data is transmitted, and received, correctly, the NFC-A tag type 1 standard uses cyclic redundancy check (CRC), more precisely, the CRC type used is CRC-16-CCITT, with registers initially set to $0xFFFF$. [17]

Formally, CRC-16-CCITT is defined with the polynomial divisor $x^{16} + x^{12} + x^5 + 1$.

In [8] this flavor of CRC is called CRC_B, it is so called to differentiate it to another CRC implementation used for NFC-A tags, called CRC_A. An important difference between CRC_A and CRC_B is that the registers are initially set to $0xFFFF$ for CRC_B, while for CRC_A they are initialized to $0x6363$.

All commands and responses, except for REQA and WUPA, use Cyclic Redundancy Check error recognition during data transfer.

2.4 Front-end and ADC

2.4.1 ADC

The ADC is Thanh Trung Nguyen's design of an integrating ADC as presented in [16], with small modifications.

Figure 2.4 shows the signals going out an in of the front-end/ADC module used in the work presented here. From the signal interface figure one can see that it is connected to the antenna coil, that is because it has its own rectifier and regulators independent on the regulators and rectifiers used for the digital core, clock recovery, demodulator and modulator. The same is true for the ADC's clock recovery.

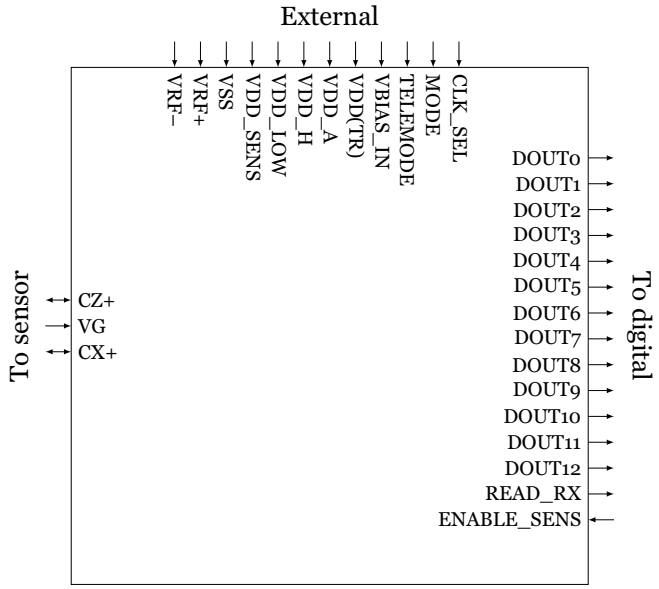


Figure 2.4: ADC top level schematic interface overview

2.4.2 Front end

The ADC's sensor connections are VG, CZ+ and CX+. CZ+ is the reference or capacitive zero connection, CX+ is the sensor connection, VG is the amplifier input. Figure 2.5 demonstrates how sensors can be connected.

Even though it is made for capacitive measurement, it can be adopted to a resistive measuring with the addition of a capacitor on the input VG in series with the resistive network consisting of Rsens and Rref. The gain of the amplifier can be controlled by selecting the input capacitance.

$$A = \frac{C_{in}}{C_F}$$

Where the internal C_F is 4pF as can be viewed from the paper [16], and in Figure 2.5 .The capacitive sensor arrangement is what is used in this work.

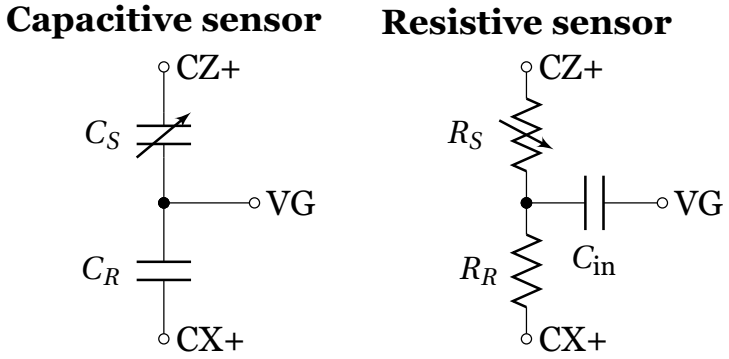


Figure 2.5: Capacitive and resistive sensor ADC connections. Subscript S denotes sensor, R denotes reference

2.4.3 ADC interface

The ADC operates continuously, as long as there is clock on the input. This is exploited in this work by adding logic for an ENABLE_SENS signal, which, when high, enables the clock input to ADC. As this is a new addition, and not present in previous work, it is explained further in 4.1.

The digital outputs seen in Figure 2.4, called DOUT<0:12>, hold the sensor value converted to digital values.

2.4.4 Converted data ready signal

After an analog to digital conversion the ADC enables a data ready signal, called *READ_RX*, output indicating that its data outputs have valid values. This signal goes low based on its internal clock without any acknowledge from the digital core interfacing it.

Do to the ADC's implemented logic, data ready signal going high do not necessary mean that data is, in fact, ready. There are two things to look out for: first, data ready is pulsed almost immediately after starting a conversion. Second, do to a glitch in the logic there is an extra data ready signal pulsed after the ADC has reset its internal counters which means that the data on the output is nothing but zeros.

Another reason for not using just any data ready signal is because the ADC is more accurate after a few cycles, which Trung explains in [16]

A timing diagram demonstration the reset after data ready signal and after starting the ADC readout procedure can be seen in Figure 2.6.

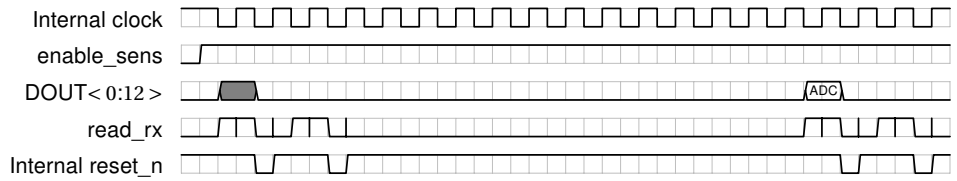


Figure 2.6: Timing diagram, from simulated VHDL model, demonstrating ADC readout procedure. Demonstrating the out of place reset occurring after a data ready signal (read_rx), and why care needs to be taken to ensure data is correctly read.

2.5 Power harvesting

To make use of the power from the carrier wave generated by the NFC reader a rectifier rectifying the AC signal to DC is required.

In [15] Trung Nguyen uses two rectifiers derived from [11, 20]. Ali Zaher proposes to use the same setup of a rectifier in [21] and a similar topology is proposed in [22] using three cascaded rectifiers.

Figure 2.7 shows the schematic of a single rectifier element as Nguyen and Zaher have utilized in their designs. This design can be cascaded in multi stage configuration to generate higher voltages on its output.[11]

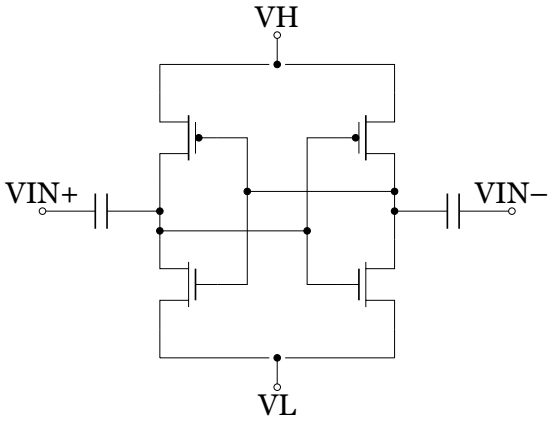


Figure 2.7: Single voltage rectifier element schematic.

2.6 Clock Extraction

The carrier wave's frequency generated by the NFC reader is 13.56kHz±7kHz [3]. To recover clock from this carrier wave Zaher propose in [22] a simple clock extraction circuit consisting of a comparator. This design can be seen in Figure 2.8.

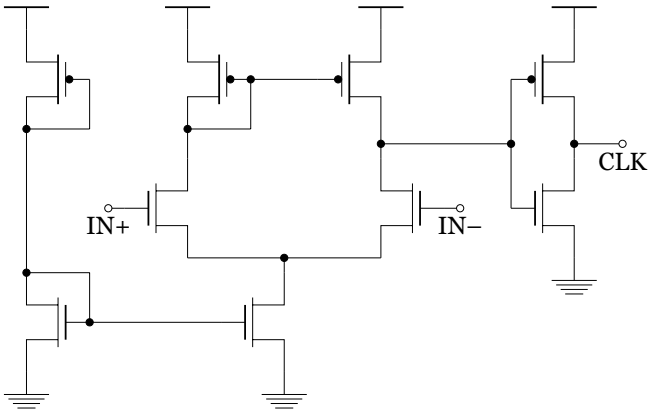


Figure 2.8: Clock recovery circuit

A similar implementation is demonstrated in [14, 15] as the chosen clock extraction implementation.

2.7 Demodulator

To interpret ASK modulated waveform from the carrier wave, a demodulator is needed. In [22] Zaher presents a demodulator which work with modulation more than 30% amplitude modulation index. The design he proposes can be seen in Figure 2.9.

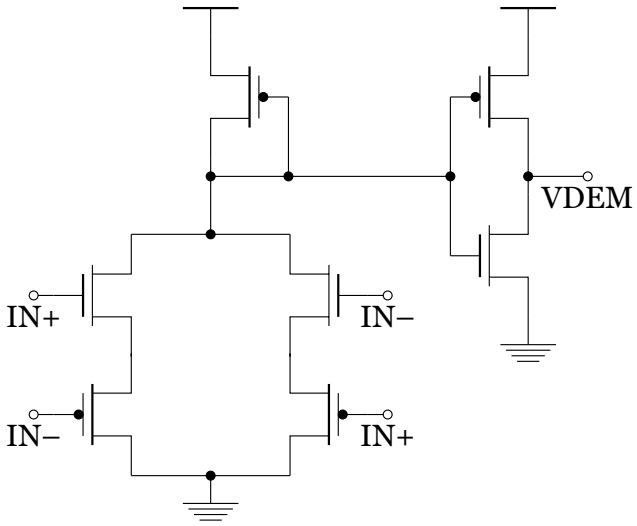


Figure 2.9: Demodulation circuit, as presented in [22]

Part II

Project

Full system overview

Figure 2.10 shows an overview of the modules and their interconnections of the system which this project implements.

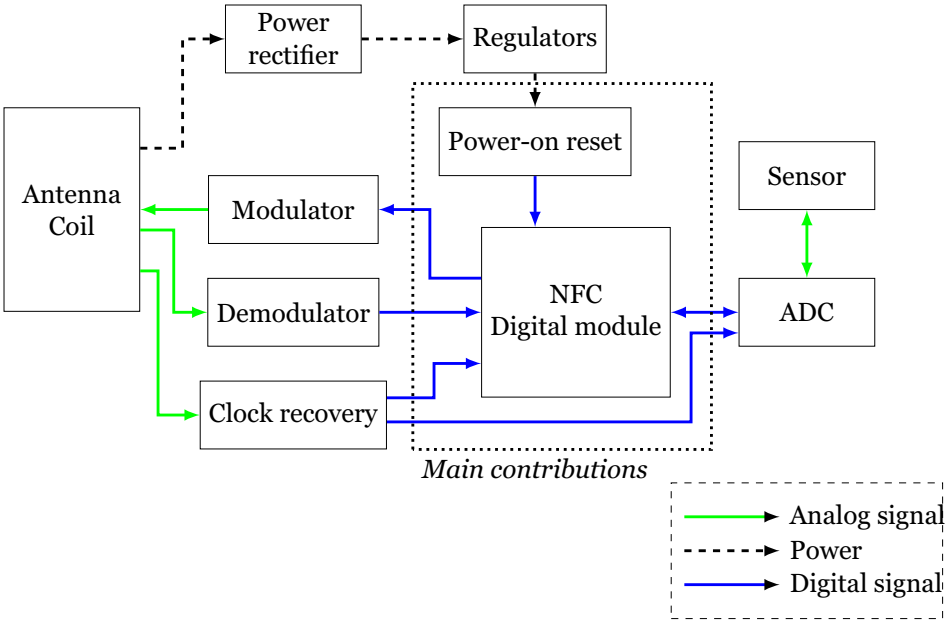


Figure 2.10: Full system module and inter-connection overview.

Chapter 3

Digital Design

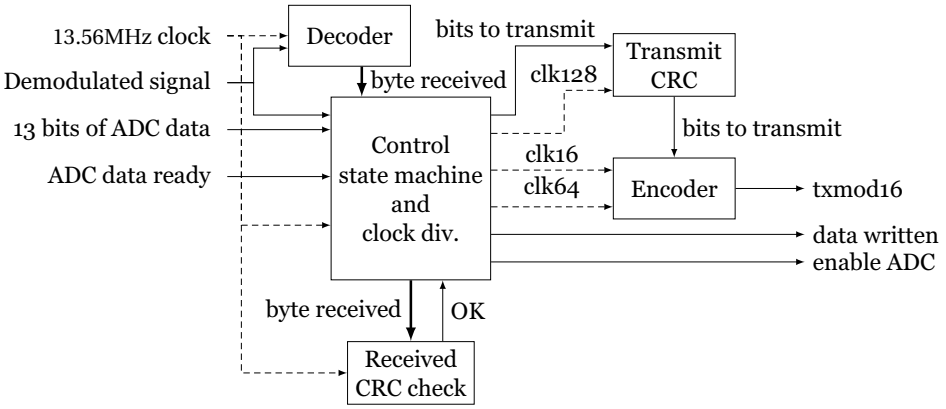


Figure 3.1: Digital core topology and important signal interconnects.

VHDL code for the digital design was written together with my supervisor Ali Zaher. We tested different approaches, where we tried each of our solutions to problems that were encountered and continued working with the best solution onto the next problem. Zaher took our solution and implemented it on a FPGA testing it in conjunction with his ASIC implemented demodulator/modulator, verifying that it could communicate with a phone acting as a NFC reader. He also did successful trials with that system connected to Trung's ADC on a separate chip.[21]

The HDL code, now tried and tested on an FPGA, was then modified further to be synthesizable into gate level logic. By employing more complex test benches, further testing and verification was done to ensure that it continued to adhere to the NFC-A tag type 1 specifications.

A VHDL model of the ADC interface's logic, ADC presented in [16], was implemented in the test bench to verify that the digital core could, in fact, operate together with the ADC as they are combined on the chip. This VHDL model allowed us to flush out bugs that would have otherwise been concealed, potentially springing forth at unfavorable moments, that is, after tape-out.

Figure 3.1 shows a somewhat simplified overview of how the VHDL code is segmented into modules, what signals are exchanged between them, and

inputs and outputs of the entire digital core. The following sections will go more into the details for each module.

3.1 Central control module

This module handles the NFC protocol logic, and includes a simple clock divider to generate clocks for uplink modulation.

It takes the decoded received bytes from the decoder module, checks CRC, determines what to do, and if it should respond. If it should respond, it creates a bit stream which is sent to the transmit CRC module, which in turn pads CRC bytes to it and sends it along to the encoder, which handles the modulation encoding of data for communication back to a reader.

The tag type which this design is based on has 120 bytes of user addressable memory, where 96 bytes, are non-volatile read/write memory available to the user.[17] The design presented here does not use any non-volatile memory, instead a static implementation of the dynamic memory defined in [17] was hard-coded in the control module. The resulting memory map can be seen in Table 3.1.

The Header ROM(HR) bytes identify what specifications are implemented in the tag. HR0= $0x11$ identifies the tag as a Type 1 NDEF capable tag with static memory map.[9] According to the specifications [9, 17], HR1 is reserved for internal use and shall be ignored.

Byte number 1 in block 2 contains the lower byte of the previously read ADC value. It can only be read using the *READ ALL* command, normal *READ* commands adhere to the setup listed explained more in the upcoming Section 3.6.

A simplistic state diagram shown in Figure 3.2 describes how the system is ready to receive and act on new commands after having received a *REQA* or *WUPA* command. The tag returns to the *IDLE* state when reset signal is pulled low, which happens during tag power up.

3.1.1 Clock divider

The NFC standard requires that the tags response times are within small time margins. More specifically: The Topaz specifications[17] require that the response time from command send completion to the tag response (Device Response Delay) is $\frac{128n+84}{f_c}$ for commands ending with a 1 bit and $\frac{128n+20}{f_c}$ for commands ending with a 0 bit. Where f_c is the carrier frequency and n is defined as 9 for the commands *REQA*, *WUPA*, *READ*, *RID*, *RALL*, 554 for *WRITE-E* and 281 for *WRITE-NE*.

To achieve this, the main control module is running at carrier frequency and generates slower clocks that are synchronized to the rising edge of input patterns and the carrier frequency.

Three clocks are generated: $\frac{f_c}{16}$, $\frac{f_c}{64}$ and $\frac{f_c}{128}$. The $\frac{f_c}{128}$ and $\frac{f_c}{64}$ clock is used for timing bit lengths and half bit lengths in the transmit module, respectively. The final clock at $\frac{f_c}{16}$ is used to generate the modulation

Table 3.1: Memory map, as it is hard coded in HDL source code. ADC value in block 2 byte 1 is reset to/default 00, but after a sensor read event it stores a byte of the previously read ADC value.

	Block	BYTE number							
TYPE	No.	0	1	2	3	4	5	6	7
HR		11	4C						
UID	0	0F	0F	0F	0F	00	00	00	00
DATA	1	E1	10	0E	00	00	00	00	00
DATA	2	00	ADC	00	00	00	00	00	00
DATA	3	00	00	00	00	00	00	00	00
DATA	4	00	00	00	00	00	00	00	00
DATA	5	00	00	00	00	00	00	00	00
DATA	6	00	00	00	00	00	00	00	00
DATA	7	00	00	00	00	00	00	00	00
DATA	8	00	00	00	00	00	00	00	00
DATA	9	00	00	00	00	00	00	00	00
DATA	A	00	00	00	00	00	00	00	00
DATA	B	00	00	00	00	00	00	00	00
DATA	C	00	00	00	00	00	00	00	00
Reserved	D	00	00	00	00	00	00	00	00
LOCK/reserved	E	01	E0	00	00	00	00	00	00

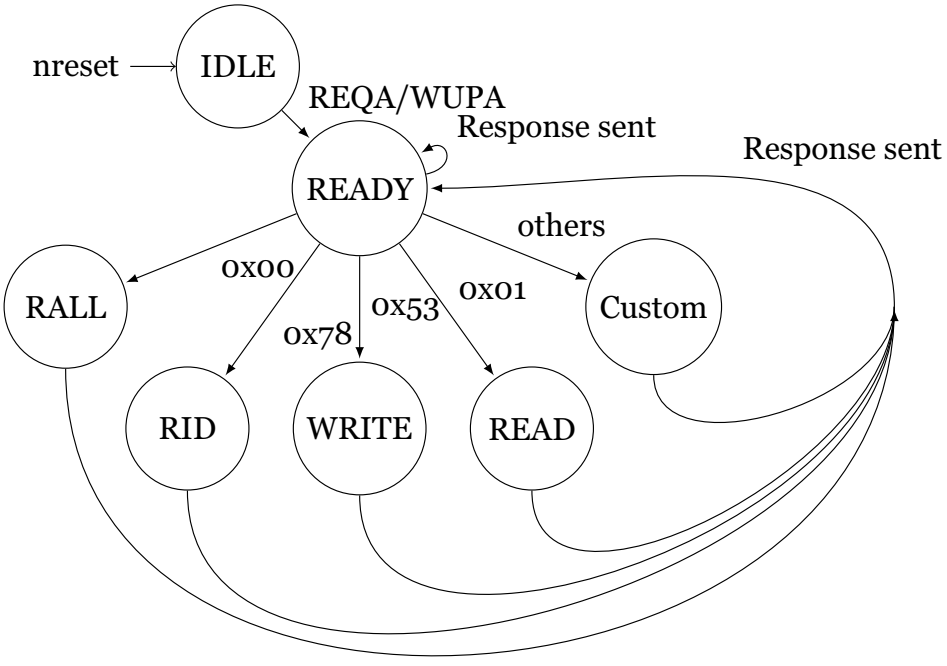


Figure 3.2: State diagram describing how commands received change the tag's operation and that after each response it returns to the READY state.

frequency used for load modulating the carrier wave, as explained in the section explaining tag to reader communication, Section 2.3.1.

The Specifications gives a margin for error of 6.5 clock cycles [17, page 11], simulation results from my implementation gives results which are within 90ns or ≈ 2 clock cycles off, which is within these margins.

3.2 Receiving Cyclic Redundancy Check

This module is relayed bits received in the receive module via the control module. It runs cyclic redundancy check (CRC) of type CRC-B on the bits. If in the end the two bytes in the CRC registers are zero the CRC code for received bits are valid. This is possible due to the nature of CRC-B, which uses xor operations and bit shifting to verify received frames.

3.3 Transmit Cyclic Redundancy Check

This module operates similarly to the receiving CRC module, see Section 3.2, except that it adds its final CRC register values to the end of the frame it receives from the control module, and subsequently pass it along to the encoding transmit module.

3.4 Transmit module

A bit stream padded with CRC bits is sent from the transmit CRC module to the transmit module.

The clock divider in the control module, see Section 3.1.1, governs the timing of tag to reader communication.

As the clocks, and bit stream is controlled, what is left for this module is to handle Manchester encoding, which is described in the tag to reader communication section under background, see Section 2.3.1 .

3.5 Receiving Module

This module takes the demodulated signal embedded in the carrier wave by the reader and interprets it into bit values.

During downlink carrier wave amplitude modulation of the carrier wave, the clock recovery module has little or nothing to work with, and consequently the clock, from the tag's point of view, disappears during modulation. This means we can not use simple pattern recognition based on time-sampling the demodulated signal, since the tags reference to time is lost in the exact moment there is something of interest happening.

An oscillator could have been implemented to offload the internal clock during the pauses in the carrier wave, but it was deemed too complex a solution for a problem which could easily be circumvented.

Our solution is instead to count the time between amplitude modulated signals, that is, time between clock losses, and use this information to discern the different patterns received. From what we know about the operation of reader to tag communication, explained in Section 2.3.2, we can distinguish patterns based on the number of cycles between each carrier modulation. Based on the known specifications and simulation experimentation the number of cycles for the different possible patterns was determined. Cycles are counted between two modulations of the carrier wave, such that the pattern decoded and subsequently the interpreted bytes are one period behind modulated bit patterns. A table covering the cycles classifying separate patterns is listed in Table 3.2. The important part to notice here is that a **Y** pattern is not recognized by it self, but from its resulting increase in pause between modulations.

Table 3.2: Cycles used in implementation to determine different patterns. A single bit duration is 128 cycles.

Cycles		Pattern	
from	to	previous	current
64	101	Z	Z
64	101	X	X
64	101	–	SoF + Z
105	189	X	Y + Z
105	189	–	SoF + X
105	189	Z	X
190	256	X	Y + X

Figure 3.3 demonstrates the state machine that decodes received patterns into their respective bit values. The bits gathered in the state machine forms a byte, which is then passed along to the control module for further processing and use.

3.6 Reading ADC data

After a glance over the memory map listed in Table 3.1, an experienced reader in the field of NFC-A tag type 1 might ask: "how do I read out the full-size sensor data, and why is it not explicitly listed in the memory map?" This section will try to answer that pressing question.

Figure 3.1, in the beginning of this chapter, describing the digital core, shows how 13 bits of ADC data is input into the control state machine, not only the 8 bits available through a *READ-ALL* command. In the implementation presented here, *READ-ALL* access different "memory space" than what is accessed using a *READ* command.

READ-ALL command access a hard-coded "memory space" with an added dynamic ADC readout byte in the middle, useful during testing. On the other hand, *READ* commands return data as they are listed in Table 3.3.

So, if we would like to read the complete ADC output data using our smart phone we need to understand how the interface to the ADC is

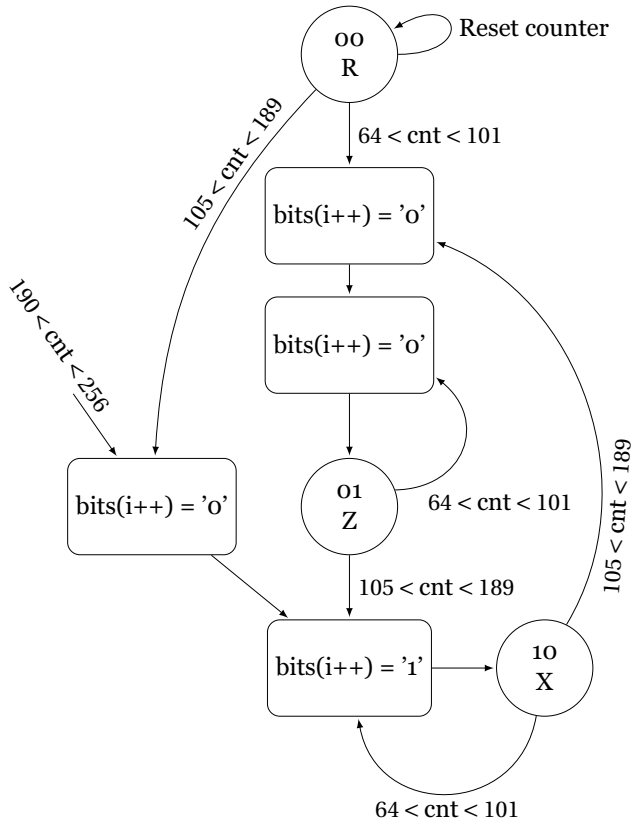


Figure 3.3: Receiving module state diagram. *cnt* is a counter counting cycles between received modulated signal lows. *i* is an indexing variable. *bits* is a register holding decoded bits

implemented. the ADC's input clock is enabled using a *enable_sens* signal. If a *WRITE-E* command is done on address $0x0D$ or $0x0E$ the ADC clock is activated, by the *enable_sens* signal going high. The *enable_sens* remains high until the ADC's data ready signal, *READ_RX*, has been high for 6144 clock cycles, ensuring that the next ADC activation starts the cycle at the same position every time.

After the tag has stored sensor values, the temporary stored values can be read using the *READ* command. The addresses listed in Table 3.3 describe which address contains data from sensor read-out.

Table 3.3: Tag's sensor data registers available to *READ* command.

Address (hex)	Block no.	Byte no.	b8	b7	b6	b5	b4	b3	b2	b1
0x0D	1	5	Lower ADC bits							
0x0E	1	6	0	0	0	Higher ADC bits				
0x14	2	4	Secondary ADC read low							
0x15	2	5	0	0	0	Secondary ADC read high				
0xXX	N/A	N/A	data in(8 downto 1), at last data ready signal							

The address setup follows the format as is listed in 3.4, although, in the implementation presented in this work, the relationship between what is read using *READ ALL* command and using *READ* command differ enough that the block and byte addresses are only listed here to indicate the implementation's heritage and specification compliance.

A full sensor read-out procedure is explained in the following subsection 3.6.1.

Table 3.4: Format of address operand ADD, from [17]

ADDR							
msb							lsb
b8	b7	b6	b5	b4	b3	b2	b1
0	block				byte		

3.6.1 Sensor read procedure

A read sensor command was implemented that causes the ADC to run and two 13-bit readouts are sampled. The command was implemented using the write-erase (*WRITE-E*) command, explained and listed in Section 2.3.3.

The readout procedure is as follows:

1. Reader initiates communication with a *REQA* or *WUPA* command to the tag.
2. The reader sends a *WRITE-ERASE* to address `0x0D` or `0x0E` (what is in its data block is irrelevant)

3. Tag enables ADC's clock, using the *enable_sens* signal explained back in Section 2.4.4.

4. The ADC signals that a conversion is complete by enabling a data ready signal, also known as *READ_RX*, when data is available at its output. On the third and fifth data ready signal the digital core samples the data and stores them in four registers holding one byte each.

   All this is happening between the *WRITE-ERASE* command is received and a response is transmitted.

5. The reader sends *READ* command to address 0x0D, 0x0E, 0x14 and 0x15 which contains the high and low bit values of the third and fifth ADC readouts, respectively.

6. The reader inverts its received values and optionally displays the ADC measured values to the user.

This procedure might be hard to digest, to simplify matters somewhat, Figure 3.4 is of assistance .

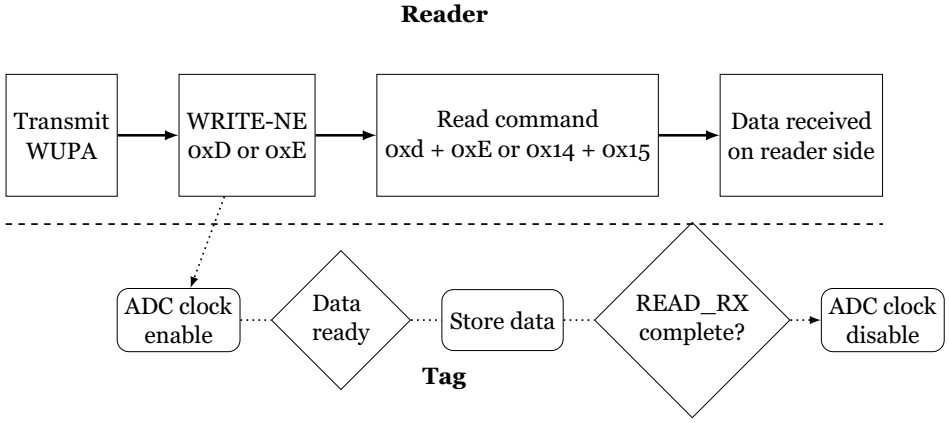


Figure 3.4: Procedure for reading a sensor using the tag and how the tag behaves in turn. *Timescale not comparable*

A diagram describing how the *enable_sens* signal is set can be seen in Figure 3.5.

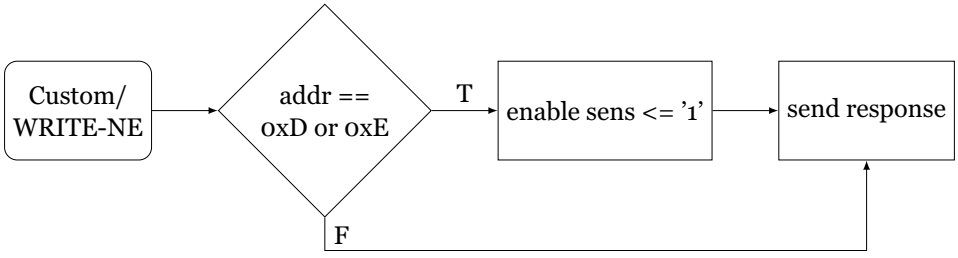


Figure 3.5: ADC clock enable diagram, as the tag sees it.

3.7 Logic synthesis

Synthesis of the digital design was done using the Cadence software package.

First the VHDL code was compiled into gate level Verilog netlist using Cadence Encounter RTL Compiler.

Next, Cadence Encounter conformal tool checked the equivalence between the synthesized netlist and the source VHDL design.

When conformity was verified, the verilog netlist was simulated in modelsim for extra verification. During verilog netlist compilation a standard delay format (SDF) file is created, it is used in conjunction with the verilog netlist in modelsim to simulate the design with an approximation of the added delays inherent in the physical implementations.

Finally, Cadence Encounter Place and route tool layout and route the verilog HDL using TSMC 90nm standard cells library made available from IMEC/Europractice.

Now we have a near transistor-level gate layout ready for implementation with the other non-digital modules. IMEC keep the transistor level design safe at their end, while we only have access to the top level blocks, but of course the delay and parasitic properties making it possible to do a synthesis are made available to us. Again it is possible to test our design against a new SDF file, now making use of the delays on the synthesized clock network and other physical layout properties. Testing using the test benches created during design creation were re-run with these new delay values, it operated within specifications. Figure 3.6 gives an overview of the synthesis process as explained.

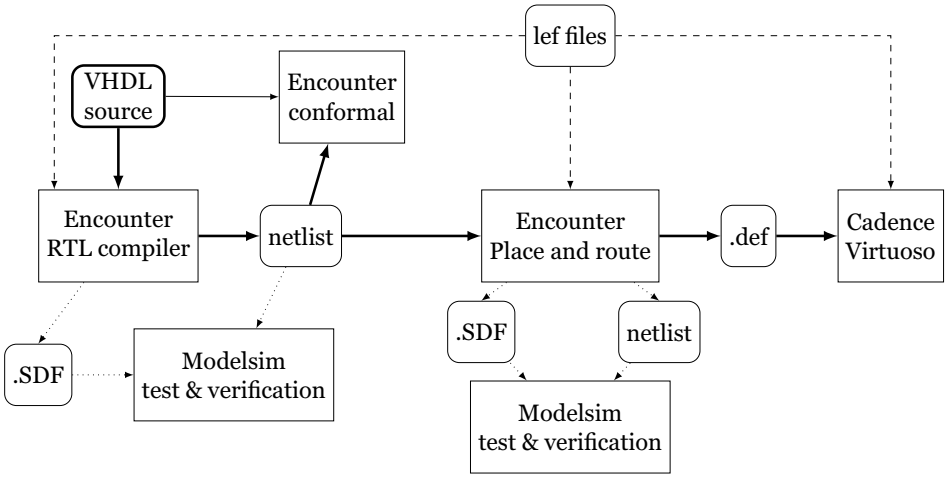


Figure 3.6: Simplified synthesis process overview

A tutorial, or walkthrough, was created for the process of synthesis; it can be seen in the appendix.

The process as listed in the appendix was based on a tutorial written by Amir Hasanbegovic and Hans K.O. Berge, and some initial research into the topic by Ali Zaher. Other aspects were learned from [19], together with

info from [4].

Chapter 4

Analog Design

The complete design consists of an ADC and front-end module, a modulator/demodulator module, clock recovery module and two identical digital cores. See Figure 2.10.

The ADC and front-end is based on the implementation shown in [16], with the modulating transistor removed, and an added enable signal which controls the clock signal into the ADC.

Modulator/demodulator, clock recovery and power regulators are based on the designs presented in [22]. The modulator/demodulator module was changed to provide a stronger clock output, and the modulating transistor was replaced with a 2.5V transistor.

As the previous work has been implemented in TSMC 90nm process, the work presented here uses the same process to enable reuse of these previously made modules.

4.1 ADC

The ADC is Thanh Trung Nguyen's design of an integrating ADC, as presented in [16], with small modifications.

An enable signal was added to control the power draw of the ADC. Figure 4.1 shows how a nand gate acts as a switch controlled by the signal "ENABLE_SENS", passing the clock signal when it is high and stopping it when low.

4.2 ADC module interconnection

The pins and interconnections for the ADC module can be seen in Table 4.1.

4.3 Modulating transistor

Communication from tag to reader is transferred using load shift keying. To change the load, a modulating transistor was implemented.

Table 4.1: ADC module terminals. External means that it is accesible on the outside of the chip, while internal means that it is only accessible from the digital core.

Pin	internal/ external	Description
CLK_SEL	external	Selects internal 50MHz oscillator or externaly connected clock on CLK_COUNT for the integrating ADC time counting signal.
CLK_COUNT	external	Clock used for counting rise and fall times in the ADC.
CLKIN_200KHZ	external	External logic clock.
TELE_MODE	external	Selects external 200kHz clock for internal logic, or if low, selects 211.88kHz clock extracted from 13.56MHz carrier wave. Here low is used.
ENABLE_SENS	internal	enables logic clock to pass through to the internal logic. Used by the digital core to enable/disable ADC operation, see Figure 4.1.
MODE	external	Selects between two modes of operation that differ mainly in how far the integrating ADC count during conversion. Here the longer count is used, which was recomended to us by the original ADC designer, Trung.
VBIAS_IN	external	270mV external bias used in ADC reference current source.
VRF+/−	external	Antenna coil input terminals.
VDD_SENS	external	voltage for sensor.
VDD_LOW	both	Internal when TELE_MODE is low. 0.4V for low voltage logic. e.g. digitizer and comperator.
VDD_H	both	Internal when TELE_MODE is low. 2.5V for analog output buffer and levelshifters.
VDD_A	both	Internal when TELE_MODE is low. 1V for analog parts.
VDD(TR)	both	Internal when TELE_MODE is low. 1V for counter circuit.
CX+	external	Capacitive sensor reference input
CZ+	external	Capacitive sensor input
VG	external	ADC amplifier input
DOUT<0-12>	both	Digital outputs for ADC. Also connected to digital core.
VOUT	external	Analog output buffer of the ADC's sampled input.
READ_RX	both	Data ready on DOUT<0-12> signal.
READ_RZ	external	Used in MODE 0. Reference value ready on DOUT<0-12>
SDO	external	Serially output ADC values.
COUNT_SET_OUT	external	Comparator and count signal out.

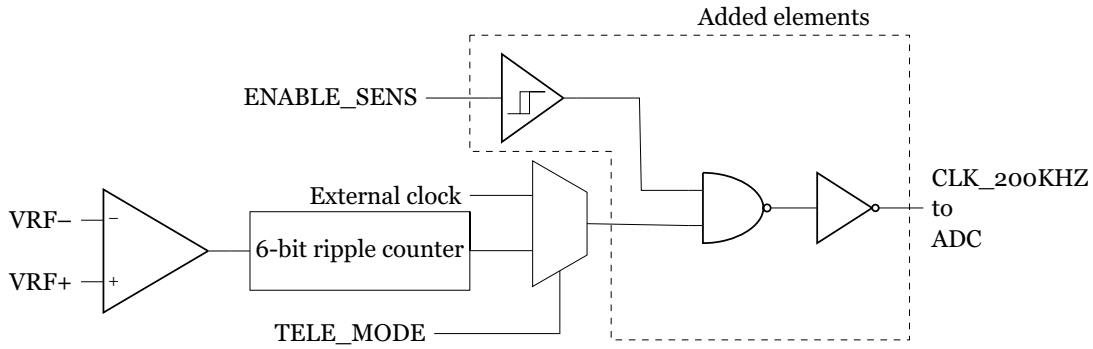


Figure 4.1: ENABLE_SENS controls whether a clock signal should feed through to the ADC or not. Note that the clock output is not 200kHz as the label implies, but rather 211.88kHz, under normal operation.

The modulating transistor modulates the impedance of the receiving coil/antenna to transmit data back to the NFC reader device. A single N-channel transistor of type “nch_25” was chosen as the load modulator. It has a length of 0.28µm, a width of 10µm, and 5 fingers.

In Figure 4.2 schematic for the modulation circuit implemented is shown.

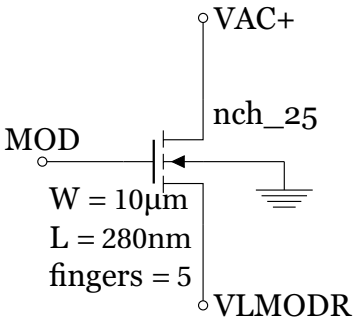


Figure 4.2: N channel transistor 2.5V version, shown with labels indicating its size and connection to the antenna coil, VLMODR is connected, through a resistor off chip, back to VAC– terminal on the coil.

Previous work, [16], have used single transistors for modulation.

The transistor strategies where simulated using the test bench shown in 4.3.

4.4 Power-on reset circuit

A power-on reset (POR) circuit creates a delayed power-on signal and it is used as a power-on reset signal for the digital module. This signal, called RESETB in the digital module, stays low for micro seconds after the operating voltage has reached its plateau of about 1V, before it too rises to the digital operating voltage. This delayed signal is used in the digital

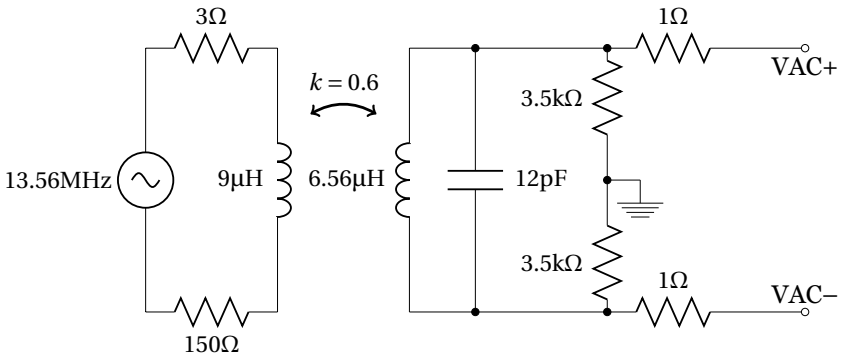


Figure 4.3: Schematic of test bench for modulation transistor

module for resetting internal states to known starting states, ensuring repeatable operation.

A power-on reset (POR) circuit was implemented based on the implementation shown in [2]. It was selected because of its low power consumption and cascadability. The schematics of this circuit can be seen in Figure 4.4. M2 and M1 length and width where chosen to balance size of layout with delay generated. A longer M1 would have made the delay somewhat longer, but simulations showed it to be only marginally so and, thus, it was kept short for simplicity.

An observant reader might notice that the m3 transistor is much wider than the minimum width, what is listed is what is implemented, although it could have been smaller without affecting the circuit's operation. It could in fact be eliminated entirely, as it is used for detecting glitches in the power supply, and thus resetting the circuit it operates on. The NFC tag presented here has little use of glitch detection, due to its longer time without power than with it doesn't need this transistor, but it is implemented in the fabricated circuit presented here.

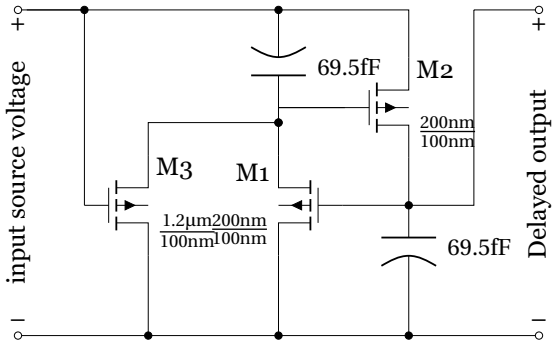


Figure 4.4: Schematic of POR delay element circuit from [2]

Four POR elements are cascaded with a buffer on its output creating the POR circuit seen in Figure 4.5. Regulated input voltage, vdd_dig, is connected to the input on the left side and a signal appropriate for RESETB is expressed on the output after a short time delay.

Figure 4.6 shows simulation of delay for a selection of rise times varied

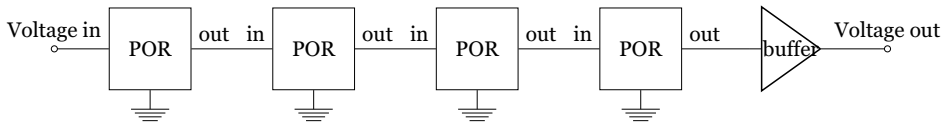


Figure 4.5: Schematic of four POR elements cascaded with a buffer to create a POR circuit

on a pulse generator on the input for the POR elements. The figure shows that our POR circuit would create an acceptable reset pulse, as long as the voltage rise time is less than $\approx 850\mu s$. Relating this number to the 5ms guard time defined as the powering up time with an unmodulated carrier before a *REQA* or *WUPA* command needs to respond[8], there is still some margin to work with.

4.5 Power harvester

Power for the digital module, modulator/demodulator and clock recovery is generated using three rectifiers of the type presented in [11, 20]. They are cascaded to generate a rectified voltage which in turn is regulated down to 1V using regulators. Two similar rectifiers are embedded in the ADC module, with its own regulators.

A change from previous designs made here in the research group is an addition of a voltage clamp on the antenna input to limit the voltage swing on the modules which have direct contact with the antenna input, for example: modulator, demodulator, clock recovery and rectifiers. It is similar to the ESD protection on the other pads, but allowing a higher voltage swing around ground. It was created from p-channel diodes clamping the voltage of the antenna near ground. Figure 4.7 shows a schematic overview of how the p-channel diodes are connected to the antenna terminals.

4.6 Voltage regulator for the digital core

Both in Ali's and Trung's design simple voltage regulators were used to generate stable operating voltages from the higher and less stable rectified voltage from the rectifiers.

An extra voltage regulator is added to generate the voltage needed for the digital core. In Figure 4.8 the schematic of this added voltage regulator can be seen. It differs from what had been used in earlier implementations by doubling the last stage into a double pmos, effectively doubling its width. This increases the current it can supply to the output.

A bias voltage of 0.86V is connected to the terminal IN+, while a 0.755V bias voltage is connected in to the bias terminal, this results in a stable 1V output voltage. These bias voltages are generated internally by a reference voltage generator meant to operate in the constant temperature of an

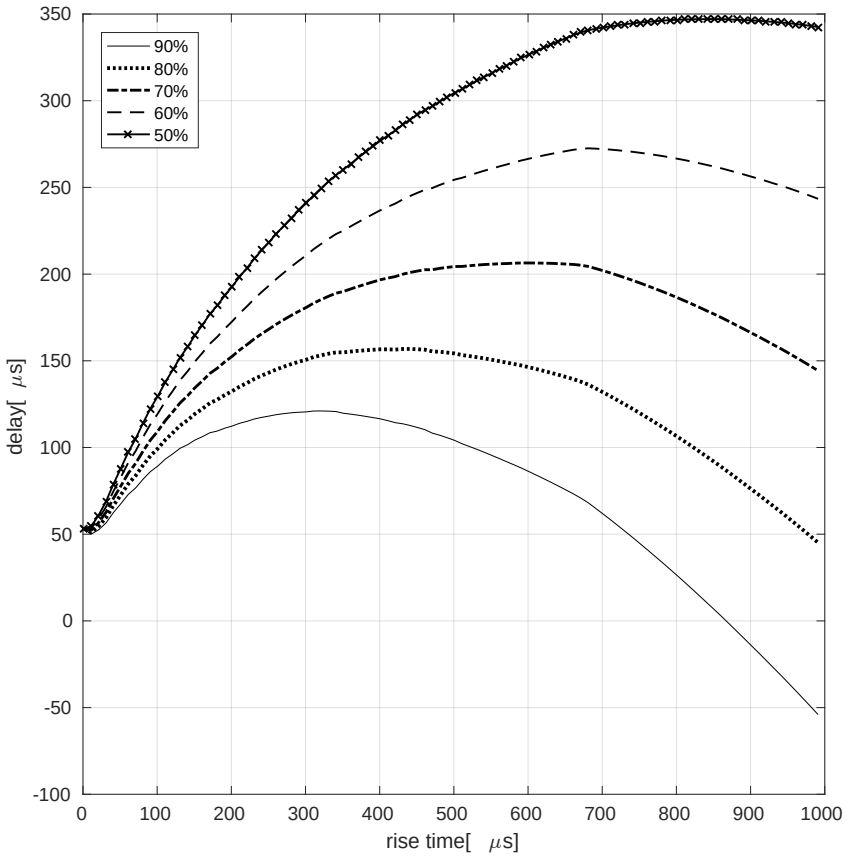


Figure 4.6: Simulated delay of four POR elements cascaded as is shown in Figure 4.5, with added components simulating how the circuit would be measured on chip. Delay is from input to output level passing their relative percentage threshold.

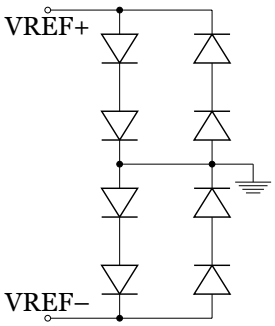


Figure 4.7: Antenna/coil input diode overvoltage protection/voltage clamp

implant, implemented by Trung in [15].

One thing to notice in the schematic in Figure 4.8 is that it generates a stable 1V output without having a 1V in as a reference. This was solved by Trung by making the differential input asymmetrical.

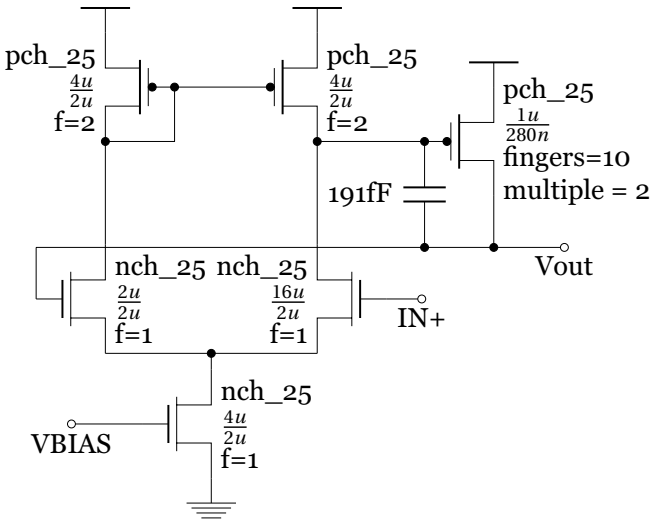


Figure 4.8: Schematic of regulator for digital core

Chapter 5

System Integration

5.1 Connecting the modules together

As the ADC and the modulator/demodulator both use the input signal originating from the receiving coil, they are directly connected on the same pads. These pads, because of their higher voltages, are connected outside the ESD protected pad frame.

Digital outputs from the ADC are level shifted up to VDD_H (nominally 2.5V); this amplifies the signal from its original 0.6V level and removes the need for an extra output buffer.

Signals originating from the ADC's output level shifters are buffered to ensure signal integrity, and protect against their higher voltage, before entering the digital core.

The digital ADC output values originating from the ADC circuits level shifters are inverted and connected to the digital module, this is done to protect the digital core and to ensure stable logic levels. This inversion also means the digital value transmitted to the phone is not directly readable as-is, but an inversion after the fact is trivial, and should therefore not give any problems.

The signals TXCMOD and ENABLE_SENS traveling from the digital core to modulator and ADC, respectively, are buffered using two serially connected inverters when exiting or entering the core. RX_IN, also known as the demodulated input signal is also buffered using two inverters in series between demodulating module and main digital core. The debugging signals sent on datawritten (#4–7) are similarly buffered on their way to die pads.

Figure 5.1 shows the finished layout with bonding pads. Size of the chip die is 1960µm × 1960µm or 3.8416mm^2.

5.2 Complete pin diagram

Figure 5.2 shows the bonding diagram for the complete chip. Each corner except the upper left, where it is the second most left pin, has an extra wire going to the cavity, pulling the cavity to VSS, ensuring there is grounding under the die.

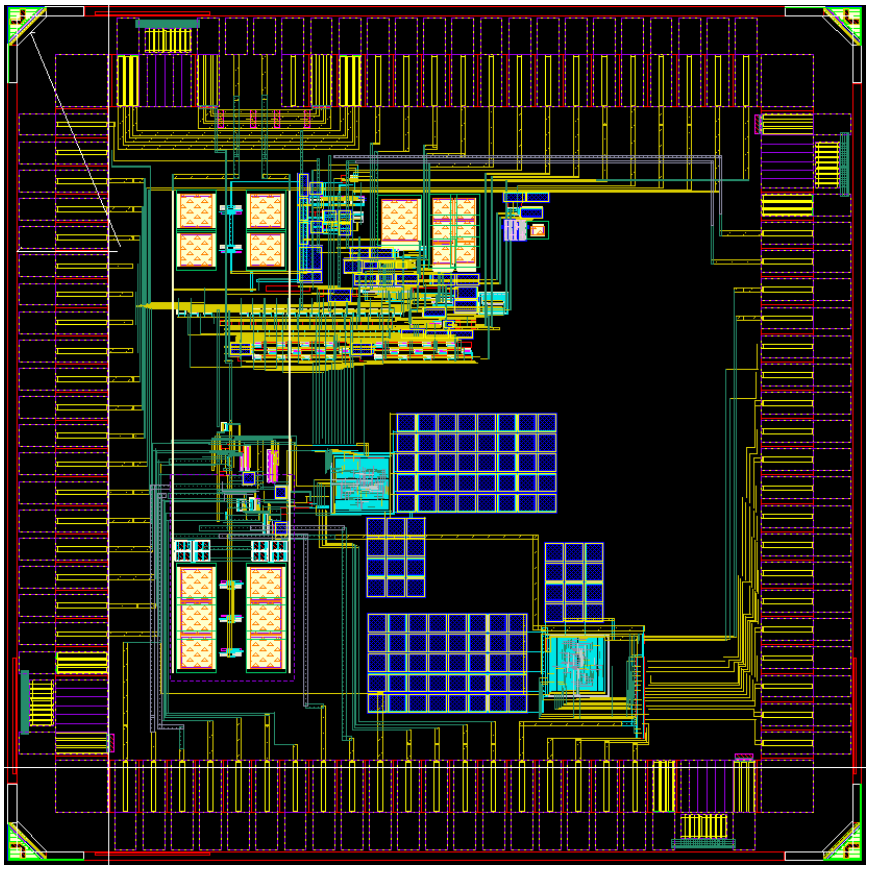


Figure 5.1: Chip layout with pad frame.

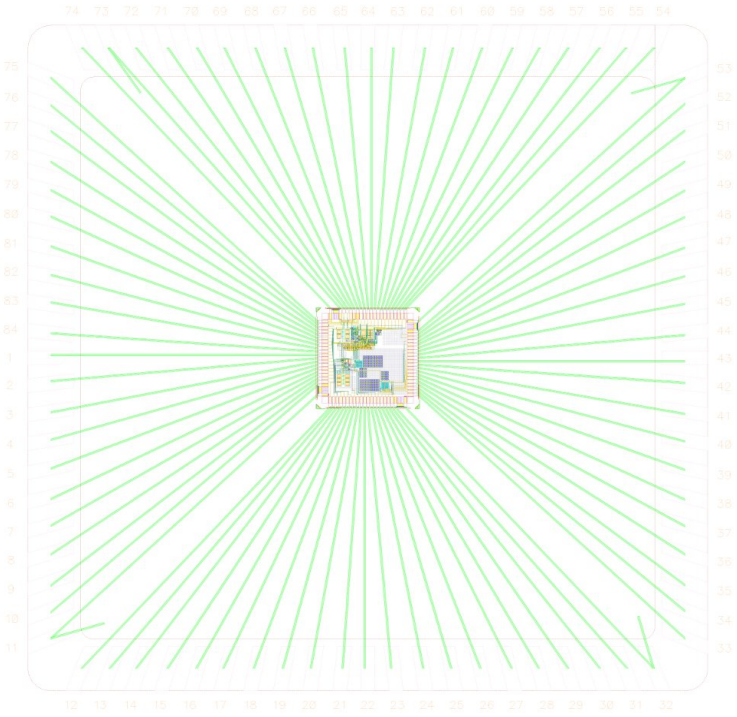


Figure 5.2: Bonding diagram

Figure 5.3 show an IO pin overview for the chip package. The pins named with suffix EX or E, and the pin named VDD_EXTRA, are pins only connected to the extra core. The extra core was added for testing the digital core without interference from the rest of the circuit.

5.3 Produced chip

Figure 5.4 show a picture of the chip die after production, the image can be compared to the layout shown in Figure 5.1, Figure 5.5 shows the die captured using an electron microscope.

5.4 ESD clamp orientation error

The pad frame of the chip has integrated ESD protection diodes connected between the VSS and VDD_H pads, or that is the simplified version of it, the truth is that we do not have the schematics of the ESD protection, nor of the pad frame, but looking at them as diodes are close to the truth. During normal conditions the diodes should be reverse biased with about 2.5V, causing practically no current draw, but for large voltage spikes the diode goes into breakdown mode, conduction current and protecting the rest of the circuit. Figure 5.6a shows how the diodes should be oriented for the pad frame ESD protection to work as intended.

In the implementation presented here, the diodes between VSS and

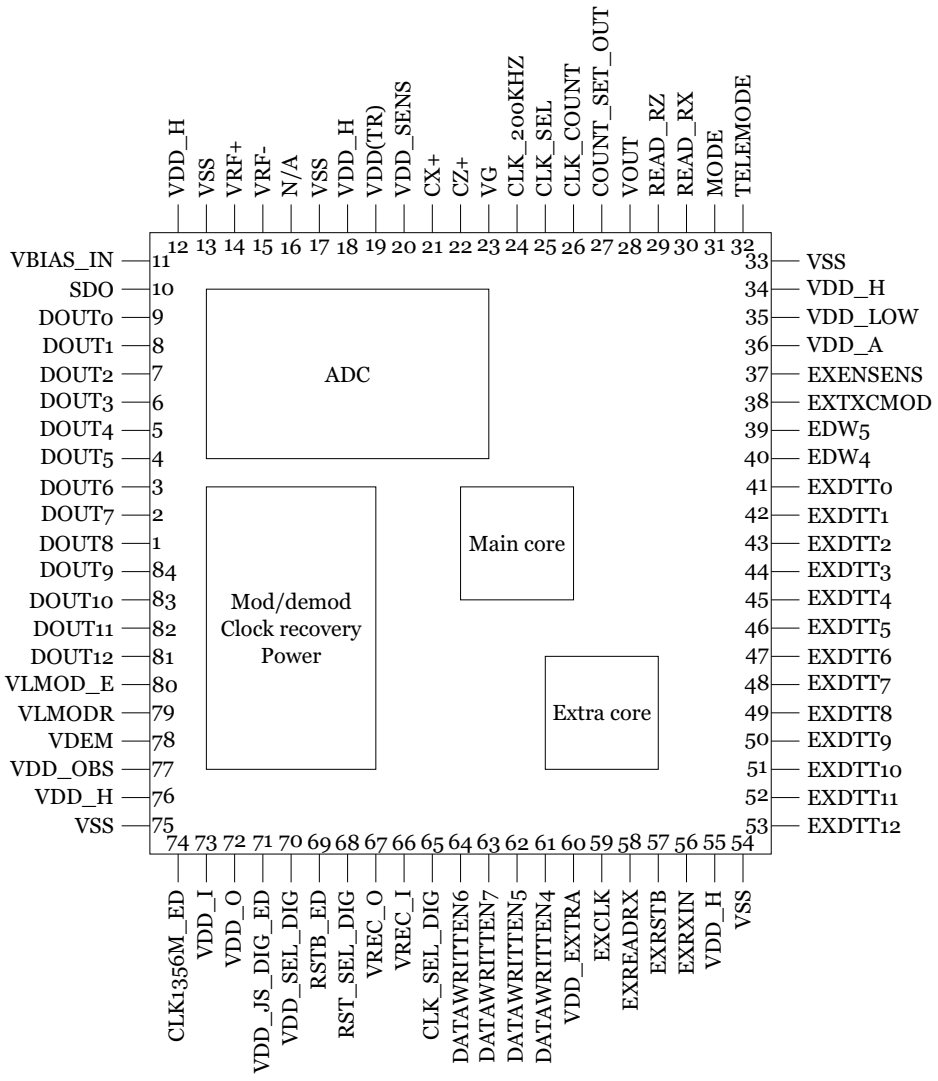


Figure 5.3: Chip package IO pin layout

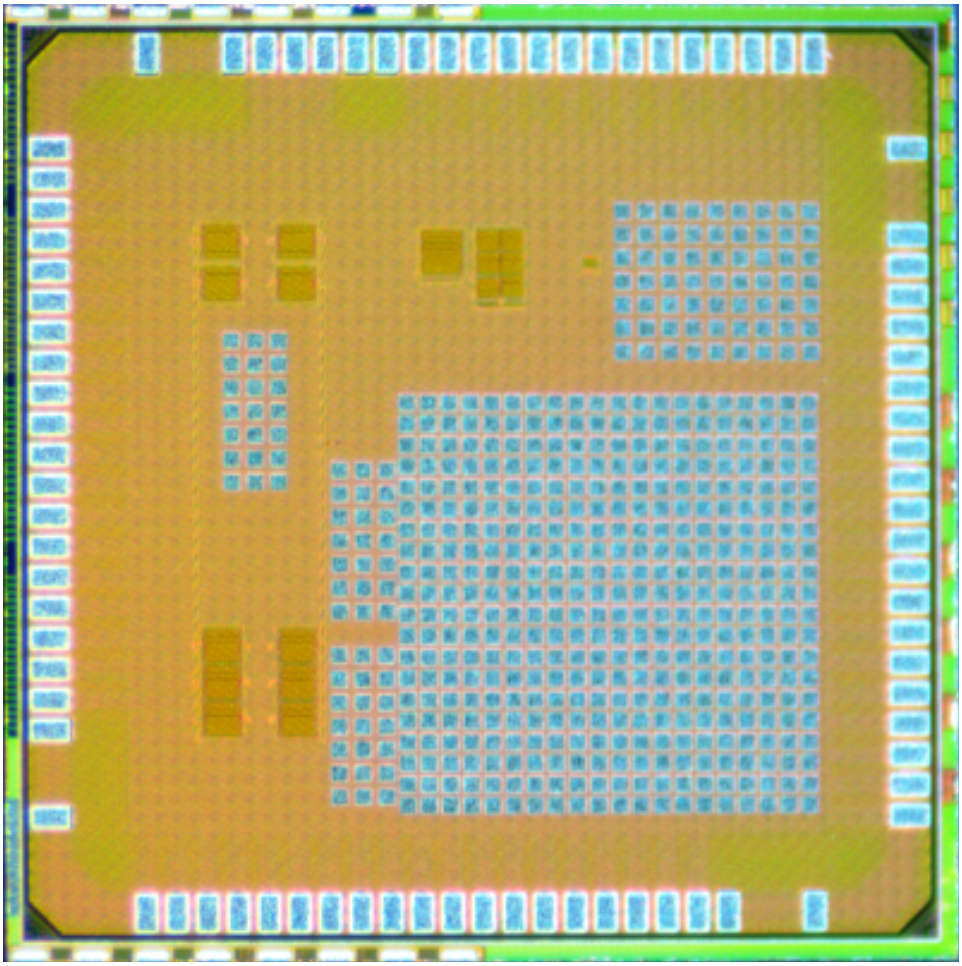


Figure 5.4: Image of chip die, captured with microscope.

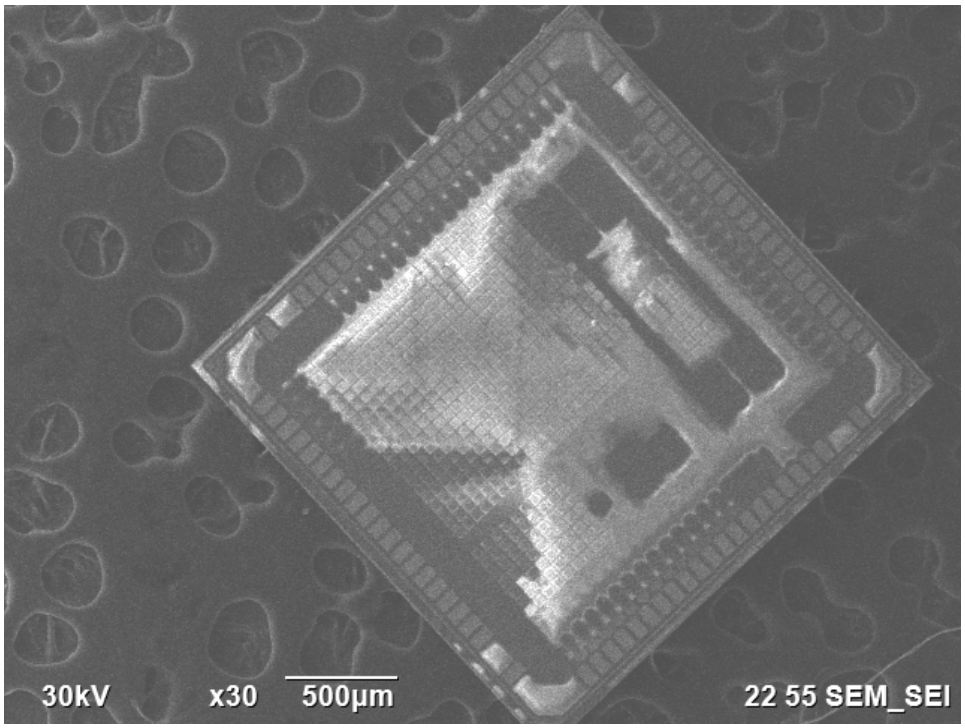
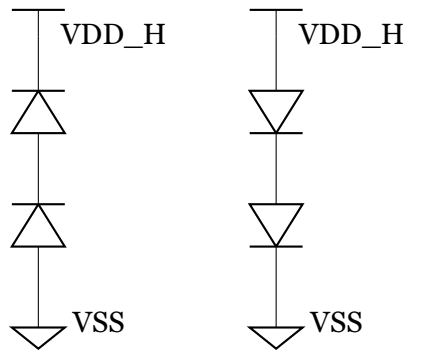


Figure 5.5: Electron microscope image of full chip.



(a) Correct ESD implementation

(b) Incorrect ESD implementation

Figure 5.6: Correct and incorrect ESD protection implementation. Simplified

VDD_H are oriented such that during normal operation the diodes are forward biased, causing them to conduct current.Some of the modules in implemented in the chip require a 2.5V voltage to operate normally, limiting the chip substantially. Figure 5.6b shows the ESD protective diode orientation as it is implemented in the produced ASIC.

This came to pass because of simple miscommunication between us and our production middle men, IMEC/Europractice, who checked the design before sending it on to the production factory. During finalization of the ASIC's layout a notification of error in the design was received from them, that said our ESD protection clamp blocks were oriented wrongly. The orientation was redone with respect to this and documentation on orientation of said blocks. It turned out that our initial rotational orientation was correct, but some of the blocks where mirrored, and this gave the reviewer an orientation warning.

5.4.1 ESD diode connection cutting

To circumvent the erroneously oriented ESD protection diodes, a fix was thought up. Namely, cutting their connection with the pad frame, this effectively removes the ESD protection. To cut these small lines a focused ion beam (FIB) machine was utilized. This solution was made available through the help and work by a PhD-student at the group, Girish Aramanekoppa Subbarao

Figure 5.7 shows ESD protection after the focused ion beam has cut the connecting metal lines; the cut goes through several layers before cutting the metal 3 layer the protection is connected with.

Because the research group here, Nano Electronics, does not have access to a bonding machine which can handle the small pitch of the pads on the die created for this project, 65µm, the cut dies where sent to an external chip to package bonder.

However, the corrected chips have not been bonded and returned to us in time to be tested before submitting this thesis. As will be reported in the measurement section, most circuits on the ASIC were simply tested with much reduced supply voltage.

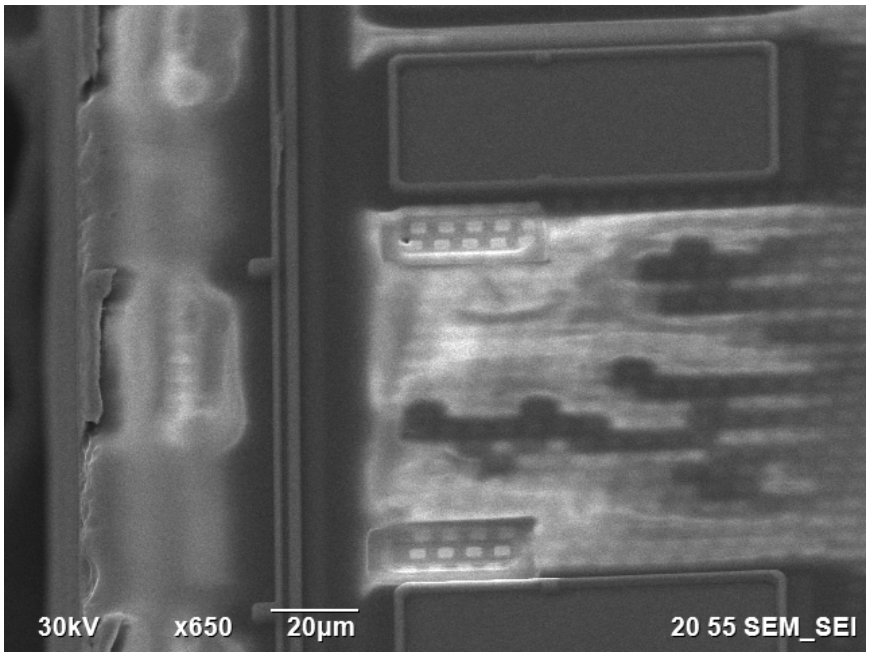


Figure 5.7: Electron microscope image of ESD diode after ion beam has cut connecting metal lines.

Chapter 6

Test Board

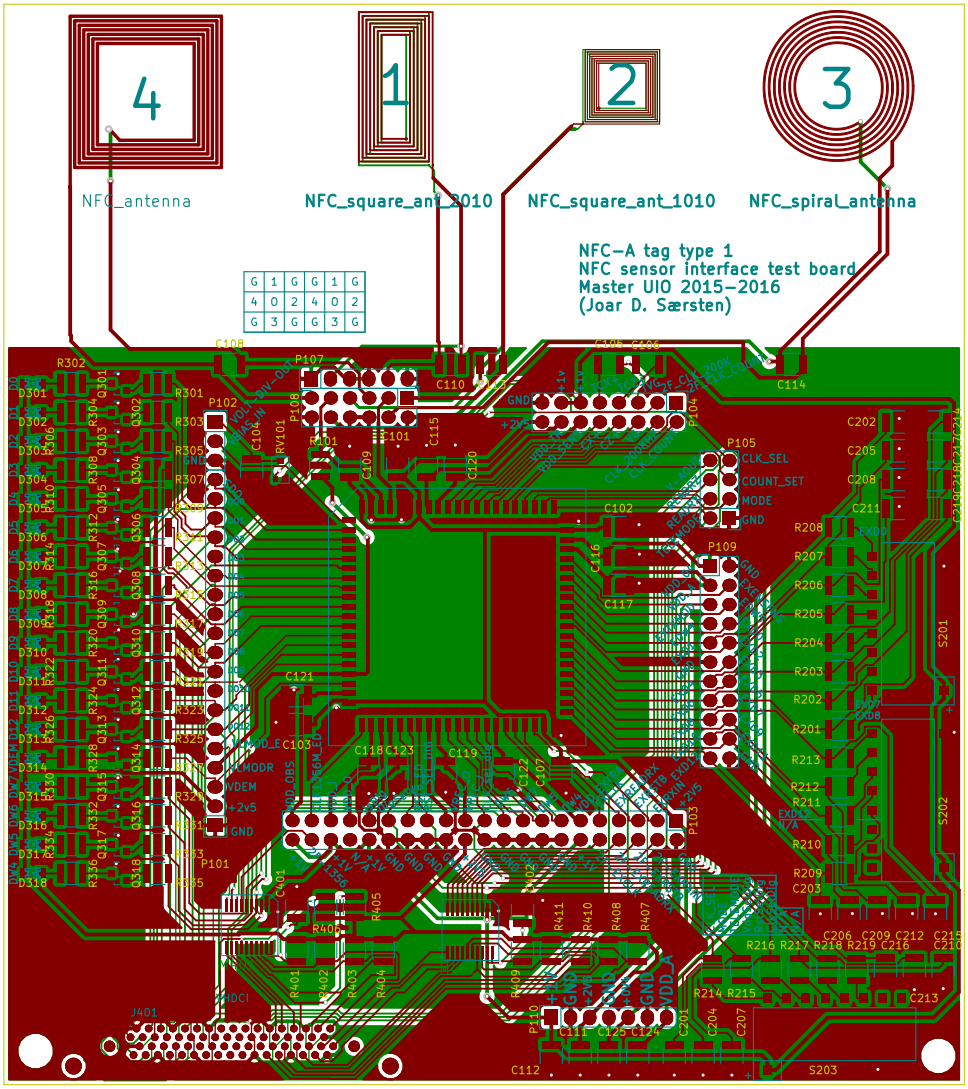


Figure 6.1: Test PCB layout

A printed circuit board (PCB) was created to facilitate the testing and/or

verification of the functionality of the chip produced.

Pin headers are connected to the chip pins to enhance the debug- and testability of the completed circuit.

The extra core's data input pins are connected to dip-switches where a digital value can be manually set.

VBIAS is created on the board with a simple voltage divider circuit. Using a potentiometer this circuit can be further used to trim the circuit if need be.

The test board was realized as a two layer printed circuit board with the dimensions 142.875mm × 127mm, Figure 6.1 shows an image of the layout produced.

A complete Bill of materials (BOM) list as it was ordered from supplier can be seen in the appendix.

6.1 VHDCI connector

The test board utilizes the Very-high-density cable interconnect (VHDCI) connector from the Digilent Atlys Xilinx Spartan-6 development board [1] for debugging and testing.

The connector is connected to the digital outputs from the ADC directly. As the FPGA board operates at either 2.5V or 3.3V, level shifters have been implemented for output pins which operates at about 1V, and an operating voltage of 2.5V for the FPGA board was chosen to coincide with the higher voltages present on the chip pins. The level shifters used are produced by Texas instruments and the model "LSF0108PWR".

The level shifters need pull-up resistors on the higher voltage side to pull the voltage up to its higher voltage. Using the following equations, based on the level shifter having a limit to 15mA, the size of the pull-up resistors where calculated.

$$R_{PU} = \frac{(V_{PU} - 0.35\text{V})}{0.015\text{A}} \tag{6.1}$$

$$R_{PU} = \frac{(2.5\text{V} - 0.35\text{V})}{0.015\text{A}} = 143.33...\Omega \tag{6.2}$$

$$R_{PU} + 10\% \approx 158\Omega \tag{6.3}$$

6.2 Indicators

A row of 18 light emitting diodes (LED) and corresponding LED driving circuits where implemented and connected to the chip internal ADC digital outputs, four of the internal core's data written signals, and the demodulator's demodulated signal. This should help during testing as no other circuits are needed to be connected to read out these values.

6.3 Mode switches

To control the operating mode and to set the digital inputs for the extra core, single pole double throw switches where used. Double throw switches where chosen since it can be used to select between two voltages to connect to one pole. In this work, they are used to connect either 1V 'high', Ground 'low' or floating to the chip terminals in question.

Six of the switches control the modes: CLK_SEL, MODE, TELEMODE, VDD_SEL_DIG, RST_SEL_DIG and CLK_SEL_DIG. While 13 switches are used for "simulating" an ADC output into the extra core.

6.4 NFC Antenna/Inductive receiver

The inductor needed to have a resonance frequency of 13.56MHz. Knowing the inductance L it is simple to calculate the needed capacitor C from the equation

$$\omega_0 = \frac{1}{\sqrt{LC}}$$

The test board has four different coil implementations, Table 6.1 list their properties.

Table 6.1: NFC inductors/antenna

Shape	Size	Layer	Turns	Inductance
Spiral	20mm diameter	Single	7	≈ 3.8µH
Square	20mm x 20mm	Single	7	≈ 3.9µH
Square	20mm x 10mm	Double	14(7x2)	≈ 5.91µH
Square	10mm x 10mm	Double	14(7x2)	≈ 1.18µH

The Spiral shaped antenna was created using Inkscape, defining its width and number of turns, and subsequently importing the design into KiCad using the bitmap to footprint converter included in the KiCad package. To simplify the double sided square antenna design process a simple script was written which takes the size and number of turns and outputs a footprint useable in KiCad.

A cell phone from manufacturer LG model L7 P700 is used as the NFC reader during testing. To facilitate the size of this phone and its NFC antenna location, the PCB was extended to give a mechanical fit to the board, as seen near the top of Figure 6.1.

To test different sizes of antenna coils an antenna selection header was created, Figure 6.2 shows the pinout of the 3×6 header pins used for antenna selection. An antenna is selected using a jumper connecting from pin 0 to one of the antenna pins ($1 \rightarrow 4$), two jumpers connected in the similar position on the other side of the header row completes the connection between antenna and chip.

G	1	G	G	1	G
4	0	2	4	0	2
G	3	G	G	3	G

Figure 6.2: Header pin description for selecting an active antenna. Position 0 is connected to the chip, $1 \rightarrow 4$ are connected to marked antennas on the board, while "G" indicates a ground connection. Using a simple jumper an antenna can be chosen during testing.

Part III

Measurements and Results

Chapter 7

ASIC Measurements

7.1 Bad ESD protection

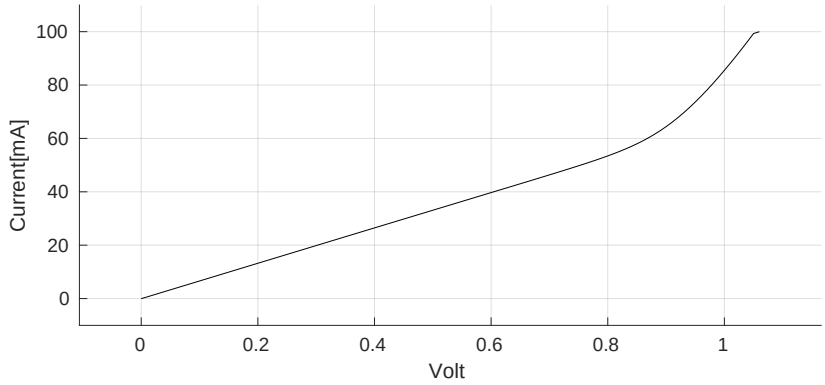


Figure 7.1: Current measured against voltage sweep on pin VDD_H, up to an instrument limit of 100mA.

A voltage sweep across the pins VDD_H and VSS measuring current draw on the ASIC without anything else connected gives the results seen in Figure 7.1. This measurement stops at 1.06V drawing 100mA because the instrument (Keithley 236) measuring current against voltage has a current limit at this value. But still, it demonstrates how the ASIC, as it is manufactured, draws a large current under normal operation, where normal operation would mean 2.5V on pin VDD_H. The current draw for 2.5V on pin VDD_H is measured, in fear of burning an ASIC.

This pin (VDD_H) is not only used for over voltage protection, for example, modulation transistor and output level shifters, the last one is incidentally also required for the digital core to read ADC data. In addition, the way the ESD protection works, also trying to protect the other pads, means that the output from the rectifier, VREC_O, and the input for the regulators, VREC_I, will have a lower resistance towards ground than a circuit without this feature.

To bypass this problem a compromise voltage of 0.85V is set on the VDD_H pin during further testing of the extra core, which the next section

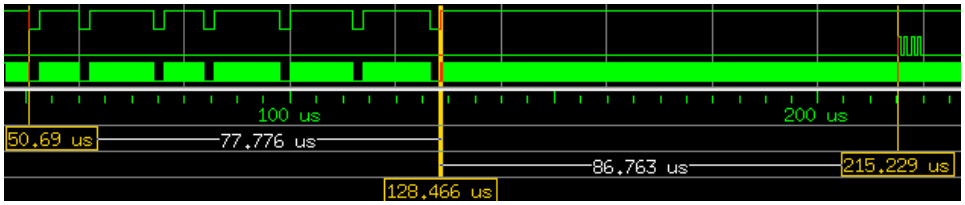


Figure 7.2: Wave diagram from VHDL simulation showing *REQA* command and beginning of response with timing. The signals are from the top: RXIN, TXCMOD, CLKIN

is about.

7.2 Extra digital core functionality

Figure 7.2 shows a simulated communication between reader and tag, the reader sends *REQA* command and the tag starts responding. This is what is recreated, but with more detail in the upcoming sections.

To test our system module by module, an extra core was added to the ASIC design. Its function is to serve as a standalone testing core that can be tested without interference from the other modules in our system. An important step to be able to test this extra digital core is to send it commands in a similar fashion that it would receive in the full system. This is accomplished here using an arbitrary waveform generator, more specifically an Agilent 33250A arbitrary waveform generator, where the points for the arbitrary waveform are generated using a MATLAB script based on NFC-A tag type 1 specifications. Figure 7.3 shows a *REQA* command waveform which was generated using a function generator's arbitrary wave function.

Using this function generator's arbitrary waveform functionality and the setup seen in Table 7.1 the response seen in Figure 7.4 was acquired. The result is close to the simulated results, as demonstrated in Figure **??** where a *REQA* response, *ATQA*, from the extra core is shown above a simulation of the same interaction.

Table 7.1: Measurement setup for REQA applied on extra core's EXRXIN and the response measured on EXTXCMOD.

VDD_H	0.85V
VDD +1V	1V
CLK frequency	13.56MHz
CLK Amplitude	1V
CLK offset	0.5V

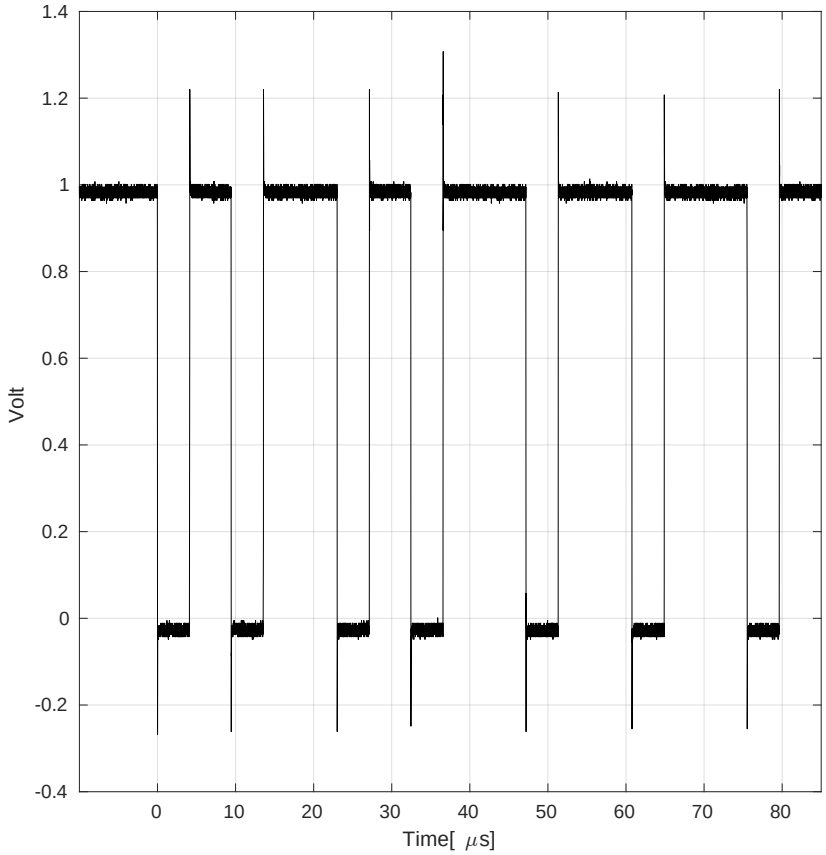


Figure 7.3: Waveform from function generator, generating REQA (0x26), that is, sequence ZZXXYZXYZ. Signal is also connected to the EXRXIN pin on the extra digital core on the produced chip.

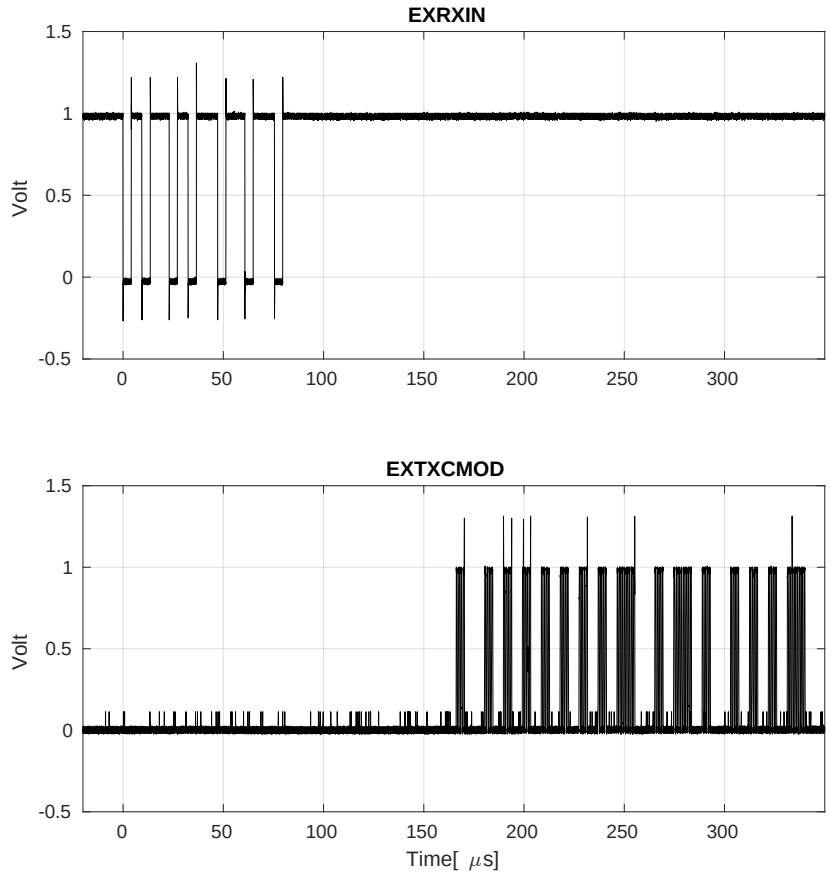


Figure 7.4: REQA command expressed on EXRXIN and the response seen on EXTXCMOD.

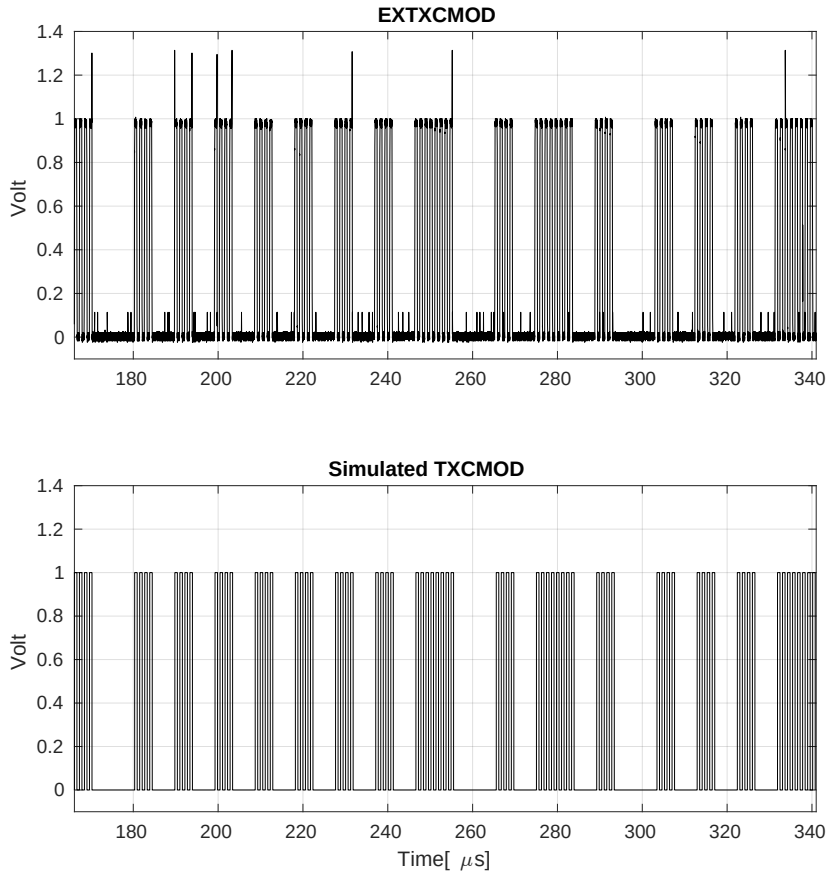


Figure 7.5: ATQA response to a REQA command, response modulation from extra core shown against simulation results

7.3 Extra digital core's current consumption

7.3.1 Static current

Table 7.2 lists some static current measurements used to determine the setup to be used for more dynamic current measurements. External data input pins, EXDTT<0:12> and EXREADRX, are pulled low, EXRXIN and EXRSTB are pulled high when no command or reset event is happening. Only the extra core's voltage supply pin's (VDD_EXTRA) current consumption is measured, other pins are driven using other external voltage sources. The two external level shifters where removed as they draw some current, affecting the operations of connected chip pins and their current draw.

An optimal setting was found for VDD_H at 0.85V, higher voltages draws more current and self-heats such that the current draw drifts in short amounts of time.

Table 7.2: Extra core's current consumption without clock signal, measured on VDD_EXTRA, other logic level pulls are handled by other voltage sources to minimize the effect of current draw from the wrongly oriented ESD protection diodes.

VDDH	VDD+1V	EXRSTB	EXCLK	EXRXIN	EXREADRX	VDD_EXTRA current draw
0.04	0	+1V(0)	(0)	(0)	0	4.123mA
0.08	1	+1V	+1V	+1V	0	2.3509mA
0.65(0.04A)	1	+1V	+1V	+1V	0	8.3361µA
0.85(0.08A)	1	+1V	+1V	+1V	0	7.8127µA
1(0.21A)	1	+1V	+1V	+1V	0	6.909µA

7.3.2 With clock

Using a 38.12kΩ resistor connected serially from voltage source to extra core's vdd input with probes on each end of the resistor. Current consumption is calculated using ohm's law $\frac{V_{source}-V_{chip}}{R}$.

The extra core has 625.4404pF capacitance internally and a 9nF capacitor connected externally, they affect the current measurements such that they are not instantaneous.

To measure the extra core's current consumption with clock input, the setup in Table 7.3 is used.

Figure 7.6 shows the current consumption for the extra core with only clock signal input. As the logic is triggered on rising edges of the clock, a rising current would be expected and is observed during low to high clock transitions.

The source meter outputs 2.81V which is divided by the resistor to 1V, it measures a current of 47.6µA, which, from the figure is consistent with the tops of the measured current consumption.

Table 7.3: Clock current measurement connection setup

From	To
Voltage source @ 0.85V	VDD_H
Keithley 236 voltage source	38.12kΩ resistor
38.12kΩ resistor	VDD_EXTRA
MODE switches	Floating position
extra core's mode switches	low (ground) position
level shifters	disconnected
Waveform generator (HP 33120A) 1V @ 13.56MHz	EXCLK
Oscilloscope (Agilent DSO6034A) probe	voltage source
Oscilloscope (Agilent DSO6034A) probe	VDD_EXTRA
Oscilloscope (Agilent DSO6034A) probe	EXCLK

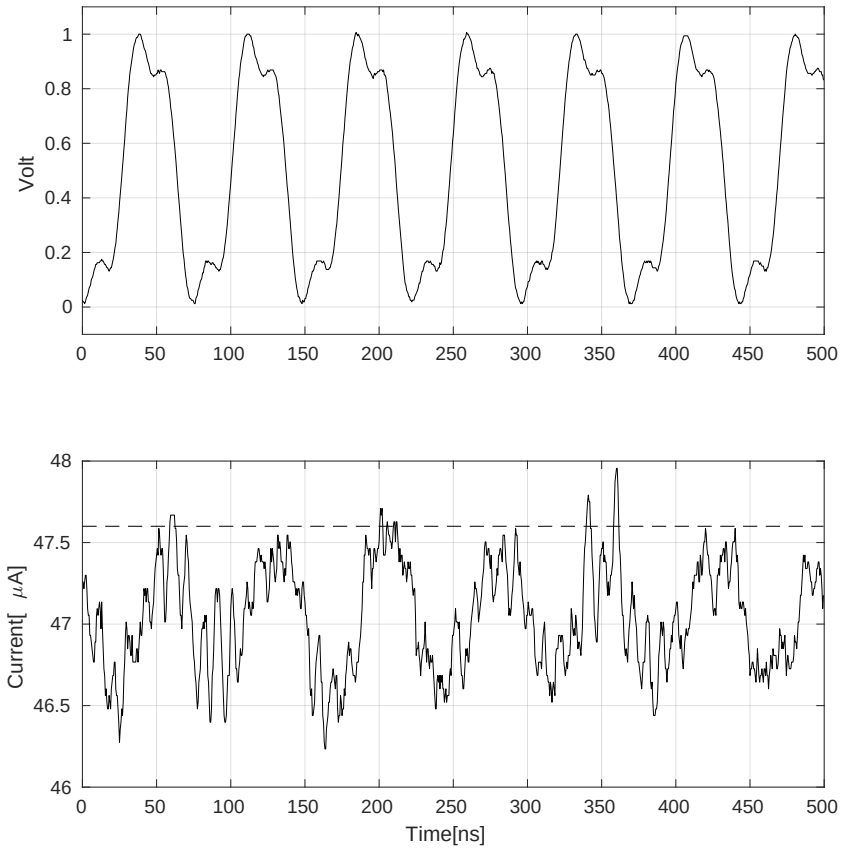


Figure 7.6: Current consumption, measured with clock signal at 13.56MHz. Showing the relationship between the clock and current consumption. Dashed line shows what the voltage source listed as current draw.

7.3.3 EXRXIN receiving REQA command

To measure extra core's current consumption when receiving a *REQA* command the setup listed in Table 7.4 is used.

Table 7.4: REQA measurement connection setup.

Continuing Table 7.3, removing EXCLK probe

From	To
Oscilloscope (Agilent DSO6034A) probe	EXRXIN
Oscilloscope (Agilent DSO6034A) probe	EXTXCMOD
Waveform generator (Agilent 33250A)	EXRXIN

Figure 7.7 shows measured current consumption during receipt of a *REQA* command expressed on EXRXIN pin, showing a small increase in current consumption. Figure 7.8 is similar but also shows the response on the EXTXCMOD pin as it was the first *REQA* after a reset event. These figures demonstrates that the digital core uses at most 50µA during a *REQA* command-receive and *ATQA*-response.

7.3.4 EXRXIN receiving REQA and subsequently RID command

These measurements use the same setup as the last sub-section, which can be seen in Table 7.4.

Figure 7.9 shows current consumption in addition to demonstrating that the extra core is responding to Read ID (*RID*) commands. A thing of note in this measurement is the lower current consumption measured here. The amount of current the extra core draws is dependent on the circuits temperature, this is not measured here, but the measurement still gives an indication of how much the circuit would draw in an ASIC where the ESD protection is not affecting the results.

RID command and response received decoded values can be read in Table 7.5.

Table 7.5: RID command and response hex values HR: Header ROM byte, CRC: Cyclic Redundancy Check, UID: Unique IDentifier

Command	CMD	ADD	DATA	UID-echo				CRC1	CRC2
	78	00	00	00	00	00	00	d0	43
Response		HR0	HR1	UID0	UID1	UID2	UID3	CRC1	CRC2
	N/A	11	4C	0F	0F	0F	0F	07	84

7.3.5 EXRXIN receiving REQA and subsequently READ ALL command

These measurements use the same setup as the last sub-section, which can be seen in Table 7.4.

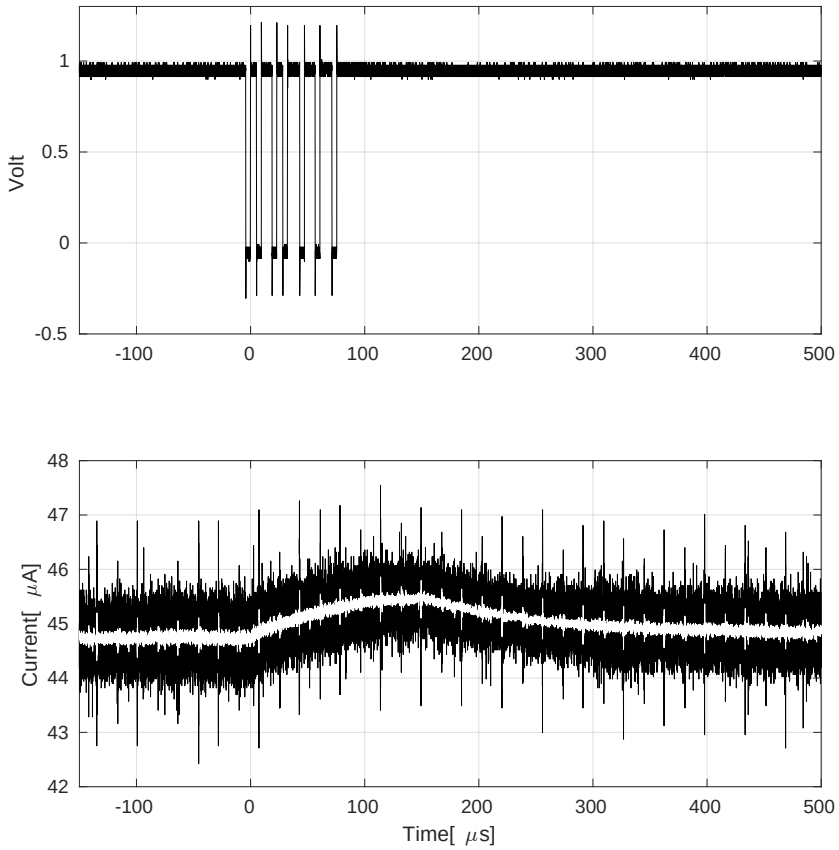


Figure 7.7: Current consumption of chip while REQA command is received. The white line in the middle of the current plot is the result of a moving average of 200 points over the wave that enclose it.

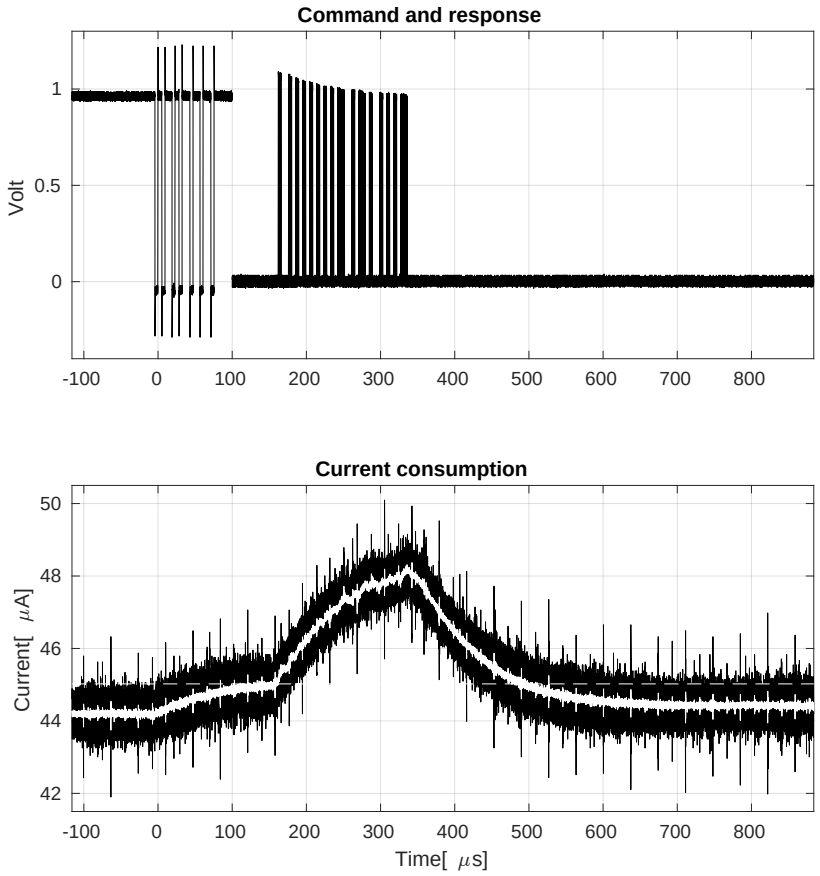


Figure 7.8: Current consumption during REQA command receipt and the extra core's ATQA response on EXTXCMOD pin. EXRXIN on top left interrupted by EXTXCMOD top right. Current consumption bottom, with dashed line showing current measured with voltage source between commands. White line in middle of current consumption is a moving average over 200 values.

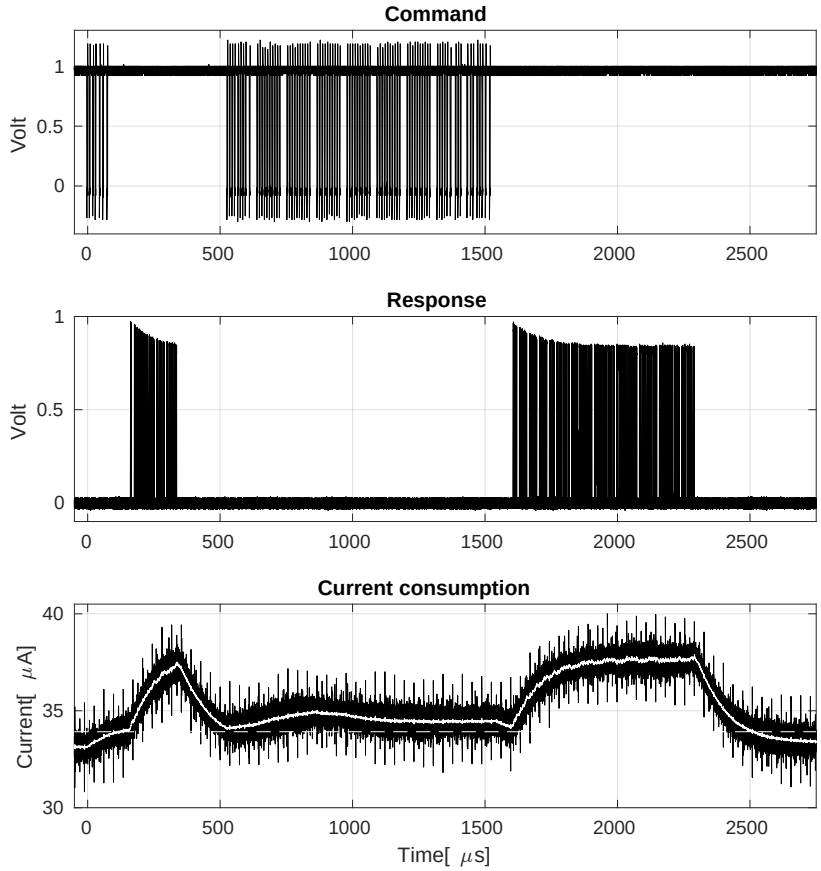


Figure 7.9: Current consumption during *REQA* and *RID* command and response output on EXTXCMOD pin. White line in middle of current consumption is a moving average over 200 values.

Figure 7.10 shows the current consumption during a *READ ALL* command, receive and response. Again, as with last section, the current measured here is low still. The expressed EXRXIN waveform includes a *REQA* command in the beginning without a subsequent response, this demonstrates how the circuit holds its state between resets and how it rejects waveforms *REQA* after entering the READY state, as described back in Figure 3.2.

As the read all command returns what is in the memory, and although the memory in this implementation is mostly static, a decoded version can demonstrate if the transfer is working as intended. To this end, a listen modulation decoding script was created in matlab, the resulting hex values can be seen in Table 7.6. Comparing this table with the Table 3.1 in section 3.1 it is apparent that the decoding script or the core's response is correct, or more likely, both are.

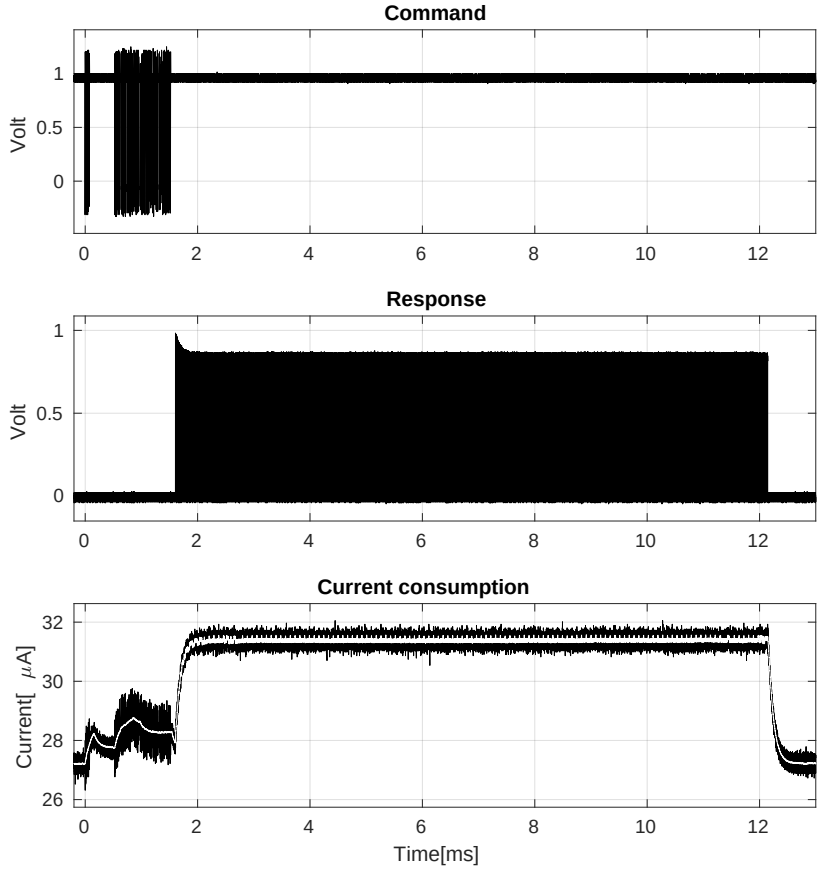


Figure 7.10: Current consumption during REQA and RALL command receiving and its *READ ALL* response output on EXTXCMOD pin. White line in middle of current consumption is a moving average over 200 values.

Table 7.6: Memory readout ordered similary as EEPROM memory map seen in [17, page 5]. First two bytes are header ROM 0 and 1, last to bytes are CRC bytes

	Block	BYTE number							
TYPE	No.	0	1	2	3	4	5	6	7
HR		11	4C						
UID	0	0F	0F	0F	0F	00	00	00	00
DATA	1	E1	10	0E	00	00	00	00	00
DATA	2	00	00	00	00	00	00	00	00
DATA	3	00	00	00	00	00	00	00	00
DATA	4	00	00	00	00	00	00	00	00
DATA	5	00	00	00	00	00	00	00	00
DATA	6	00	00	00	00	00	00	00	00
DATA	7	00	00	00	00	00	00	00	00
DATA	8	00	00	00	00	00	00	00	00
DATA	9	00	00	00	00	00	00	00	00
DATA	A	00	00	00	00	00	00	00	00
DATA	B	00	00	00	00	00	00	00	00
DATA	C	00	00	00	00	00	00	00	00
Reserved	D	00	00	00	00	00	00	00	00
LOCK/reserved	E	01	E0	00	00	00	00	00	00
CRC		E9	F9						

7.4 Reading digital inputs

As a substitute for an ADC connected to the extra core, switches are used to input data that can be read when an enable sensor and subsequent read operation is done on the addresses 0x0D, 0x0E, 0x14 and 0x15 as explained in Section 3.6.1.

The waveform generator used for sending commands to the core has a finite waveform pattern memory, that is, approximately two commands could be sent at a time from the waveform generator to the tag at a time. Luckily the long time needed to reprogram the waveform generator did not pose a significant problem because the specification for the NFC protocol [8] does not specify a maximum amount of time the tag can wait for a new command, which means, that the tag will sit idly by and wait for its next command. These time limits are called *Frame Delay Time Listen → Poll* and *Reader-Reader Data Delay*, for time between tag frame and reader frame and time between reader frames, respectively.

7.4.1 Write erase command with special address 0x0E

Measurement data from expressing a *WRITE ERASE* command, writing zeros to address $0x0E$, can be seen in Figure 7.11. The address $0x0E$ is one of the special purpose addresses listed in Section 3.6.1, which when written to enables the ADC clocks. As the plot shows, the command works as expected by pulling the ENABLE_SENS signal high when something is written to the

$0x0E$ address. ENABLE_SENS is pulled high and stays high because there are no READ_RX signal going high in this simple model, which would have signaled the core to pull ENABLE_SENS low again.

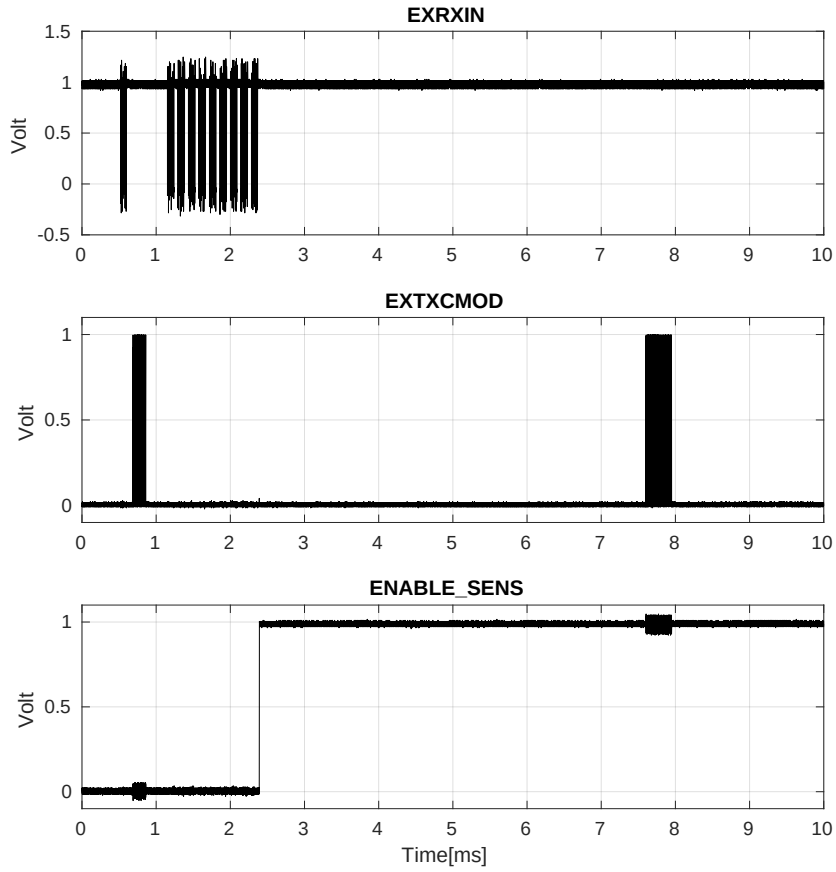


Figure 7.11: Measurement verifying the functionality of a *WRITE-ERASE* command on special address 0x0E.

Demodulating the TXCMOD output seen in Figure 7.11, we get the hex values $0E005795$. The first byte is the address, $0x0E$, the second byte is the data which in a normal tag would be what has been written, but here it just echoes what was transmitted down to the tag. The last two bytes are CRC_B bytes; a quick CRC_B calculation using the CRC_B algorithm as shown on the last pages of [17] verifies that they are correct.

Scripts were created to send two *READ* commands at a time using a waveform generator's arbitrary waveform functionality. In Figure 7.12 a demonstration of two *READ* commands reading address $0x0D$ and $0x0E$ after a simulated read sensor event are shown.

To simulate ADC output the simple solution of switches as data input and a push button for the data ready signal (READ_RX) was devised. Similarly to how the sensor read procedure is explained back in Section 3.6.1, the simulated sensor read-out procedure becomes:

1. Sliding switches are set to an arbitrary value.

2. *write erase* command on address `0x0D` or `0x0E` set the ENABLE_SENS signal.

3. The push button for data ready signal (connected to EXREADRX pin) is pressed, the switches are read by the tag and ENABLE_SENS (pin EXENSENS) is pulled low.

4. *READ* commands on addresses listed in 3.3 retrieves the read switch positions.

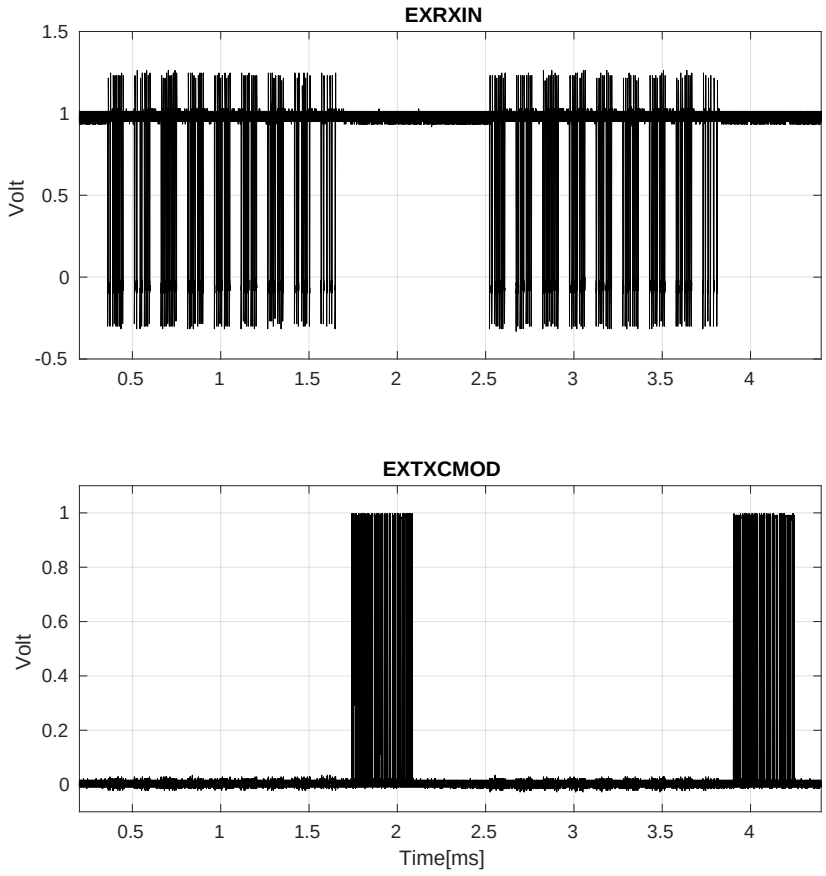


Figure 7.12: Reading out two bytes from registers holding read data input values. Here address $0x0D$ and $0x0E$.

In Table 7.7 responses to *READ* commands on the special addresses after a simulated read sensor event are listed. The data input switches, simulating an ADC output, where set to the positions $0x1E65$, which the read values match.

Description	Address	Data	CRCB1	CRCB2
Last low byte	0D	65	94	8B
Last high byte	0E	1E	A8	6C
First low byte	14	65	1D	C9
First high byte	15	1E	91	1D

Table 7.7: Read sensor read out after a simulated sensor read, switches in position $0x1E65$.

7.5 POR circuit delay measurements

Back in Section 4.4 the delay generated by the implemented POR circuit was simulated, see Figure 4.6. To check these simulations a measurement on chip is done. An arbitrary waveform generator with pulse function with adjustable rise and fall time supplies the power to the digital voltage input terminal. An oscilloscope is connected to both this voltage input and a buffered RESETB output terminal, which are VDD_JS_DIG_ED and RSTB_ED terminals respectively. Of note here is that this is done on a circuit with a bad ESD protection, which means that a voltage of 0.85V pulling about 80mA is powering the pad frame ESD protection circuit, not the 2.5V one would prefer. Other terminals that are powered are VDD_OBS, VDD_SEL_DIG, and RST_SEL_DIG; they enable the buffers and transmission gates that enable us to measure on these normally internal-only signals. Instrument connection and measurement setup can be seen in Table 7.8.

Table 7.8: Measurement setup for POR circuit delay measurements.

Instrument	DUT pin
Waveform generator	VDD_JS_DIG_ED
Oscilloscope probe	RSTB_ED
Oscilloscope probe	VDD_JS_DIG_ED
Voltage source @ 1V	VDD_OBS
Voltage source @ 1V	VDD_SEL_DIG
Voltage source @ 1V	RST_SEL_DIG
Voltage source @ 0.85V	VDD_H

Using the setup as described above, the delay time against pulse rise time measurement in the POR circuit are seen in Figure 7.13. Looking at the 50% line and comparing it to the simulated values, one can see that the measured time delay values are about twice that of the simulated ones in relation to rise time. Given that we have the 5ms power-up time, explained in Section 4.4, we still have a good margin to work with, even with our new twice-as-large delay values. The large delay enables the circuit to charge its external capacitive power storage for longer amounts of time, without the potential power draw from an active digital module.

In Figure 7.14, a sample from measuring POR delay is shown. It shows the rising edge of a square wave with a rise time of 500µs expressed

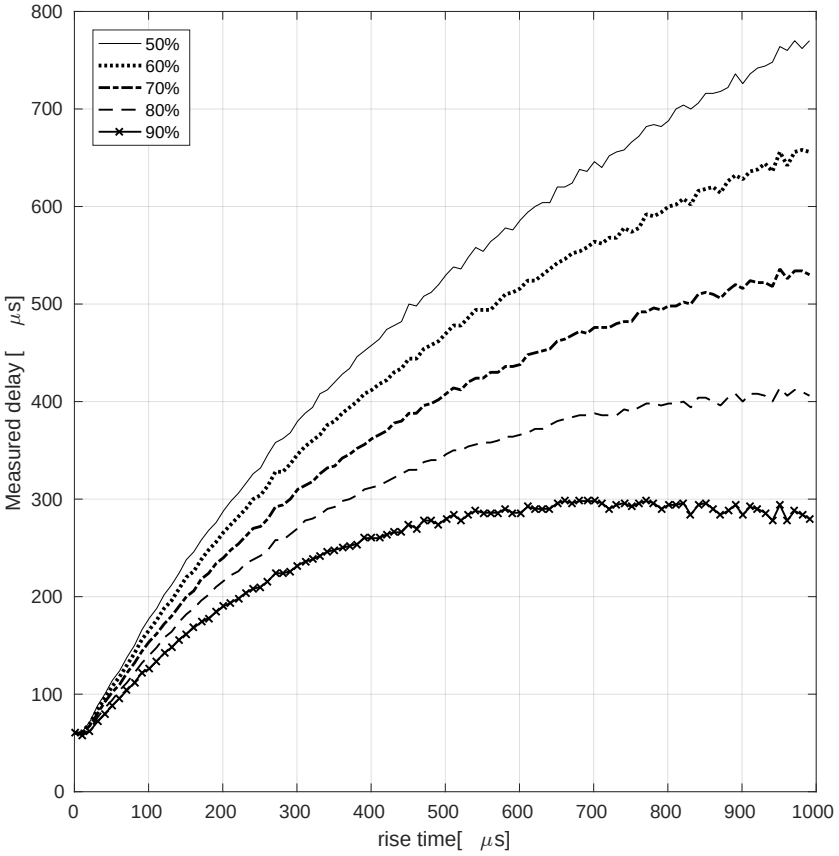


Figure 7.13: Measured delay time POR circuit in fabricated ASIC. Shows relation between delay time and rise time of an expressed pulse acting as a voltage source on digital voltage input.

on the VDD_JS_DIG_ED pin, and the delayed active-low signal on pin RSTB_ED. The RSTB_ED pin remains high until the input voltage on the VDD_JS_DIG_ED pin goes low again.

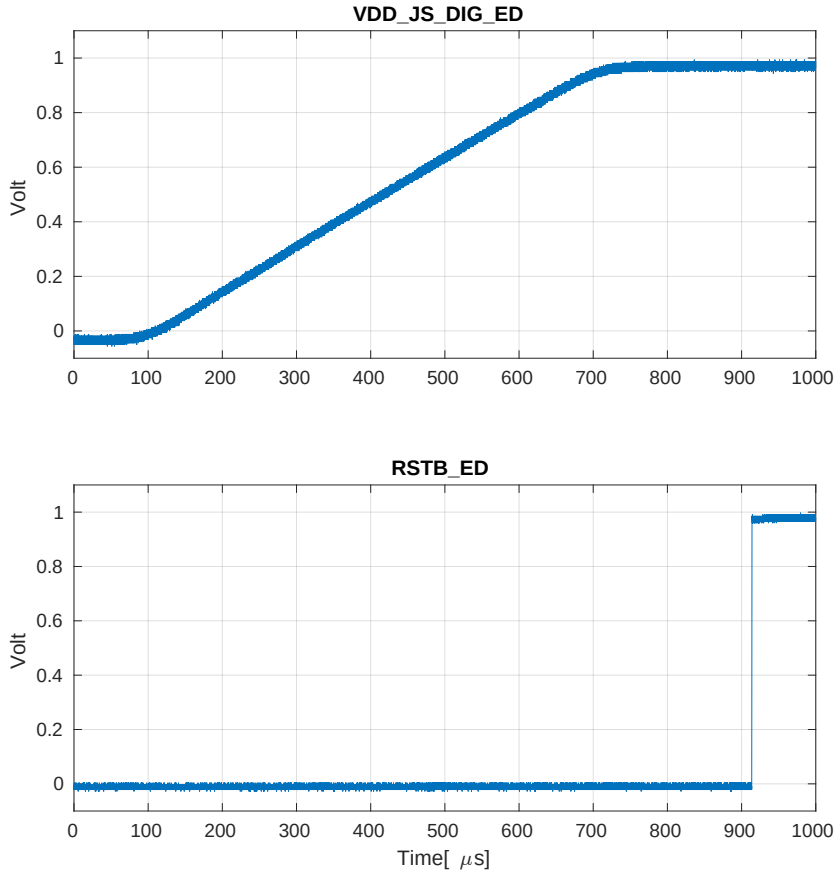


Figure 7.14: First flank of a POR delayed RESETB signal as measured on RSTB_ED. Rise time for the square wave set to 500µs, expressed on pin VDD_JS_DIG_ED. Shows how the active-low reset trails the input voltage.

7.6 Clock Extraction Measurement

The clock extraction module explained in Section 2.6 is tested using the setup listed in Table 7.9. That is, a smartphone, here of type Sony Xperia Z1, is used as an NFC reader. During the normal discovery phase of the NFC reader, it generates a carrier wave to power up tags that are in its vicinity before sending any commands. This power up phase is used here to test the clock recovery module, and in return a recovered clock on the output is observed.

The results seen in Figure 7.15 demonstrate the clock extraction module working, when it is powered externally. The frequency on both the antenna

Table 7.9: Measurement setup for clock recovery.

VDD_H	0.85V
VDD +1V	1V
VDD_OBS	1V
VDD_I	1V
CLK_SEL_DIG	1V
VRF±	Antenna 4
Reader	Sony Xperia Z1
Osciloscope probe	Antenna
Osciloscope probe	CLK1356M_ED

and on the clock output is measured at 13.560136MHz, which is within the margins set in [3] of ±7kHz, as explained in Section 2.6.

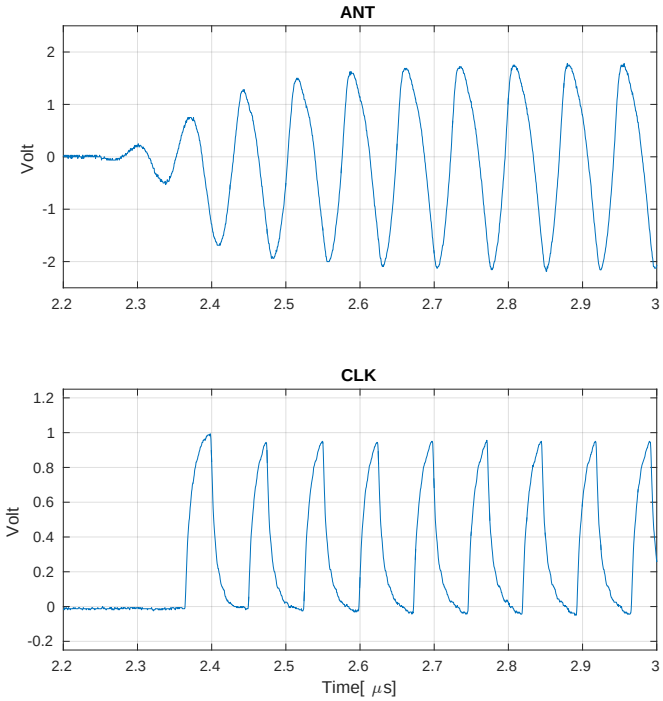


Figure 7.15: Clock extraction from induced wave from an NFC reader with external power for clock recovery and buffers.

7.7 Testing the Demodulation Module

The circuit explained in 2.7 is tested using the same setup as used for testing the clock recovery module, listed in Table 7.9, except for an added probe on the VDEM output pin. Figure 7.16 shows measurement results for a test of the modulation circuit where the signal it demodulates is from the discovery

wave an NFC reader sends to know if there are any NFC enabled tags within range. Comparing the demodulated signal VDEM with the *REQA* command generated from specifications in a waveform generator, seen in Figure 7.3, it is apparent that this signal is a *REQA* command too.

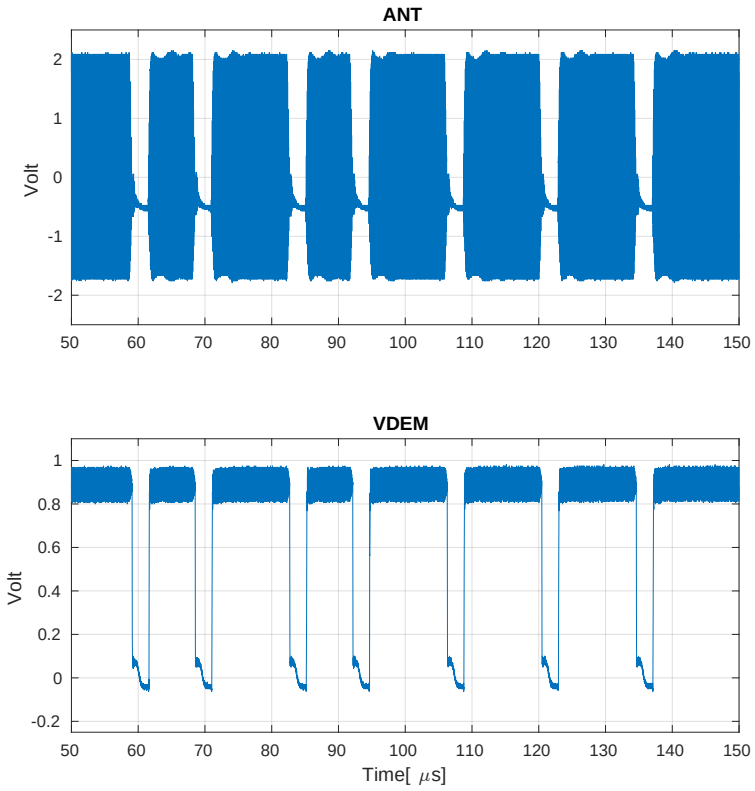


Figure 7.16: Recovered antenna carrier wave and demodulated signal. Demonstrating the functionality of the demodulation circuit. That is demodulating a signal from a smart phone acting as a NFC reader.

7.8 Extra core connected to Demodulation and clock recovery

Looking to the preceding measurements, the next logical step is either to test the complete system, or, as is done, connect what has been tested together with the extra core.

Again using a setup quite similar to what is used for testing the clock recovery module, except this time the phone used is a LG P800, and the signals probed are the extra core's modulation output, EXTXMOD, and again demodulated signal. Table 7.10 lists the setup.

Figure 7.17 shows the observed waveform that first the NFC reader, here a LG P800 smartphone, express on the carrier wave, seen in the demodulated signal of VDEM. And then the response from the extra core,

Table 7.10: Measurement setup for extra core connected to demodulation and clock recovery.

VDD_H	0.85V
VDD +1V	1V
VDD_OBS	1V
VDD_I	1V
CLK_SEL_DIG	1V
VRF±	Antenna 4
NFC Reader	LG P800
EXRST	external button
Osciloscope probe	Antenna
Osciloscope probe	EXTXMOD
Osciloscope probe	VDEM
VDEM	EXRXIN
CLK1356M_ED	EXCLK

seen in the signal EXTXMOD, so called as it is the extra core's transmission modulation output pin.

From our previous excursion into demodulation of a signal from an NFC reader, we are now easily aware that the waveform represents a *REQA* command.

Now the transmission modulation signal might not be as easy to read, therefore the decoding script mentioned earlier is used to decode it; it returns the hex value $000C$. Back in Section 2.3.3 this response is explained.

The extra core was not connected to a modulation circuit, so the waveform seen is the raw output from the extra core's transmission modulation output, EXTXCMOD.

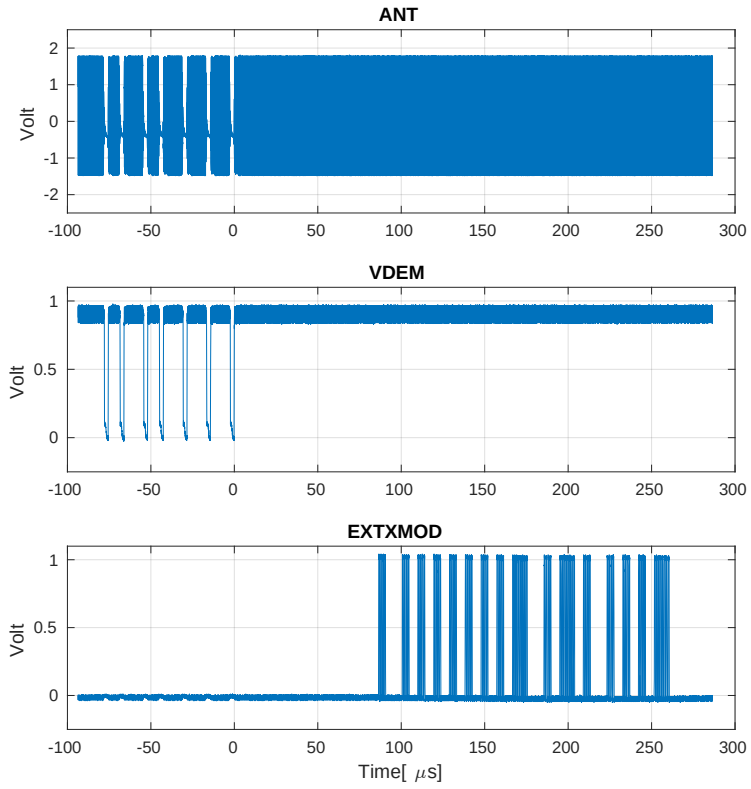


Figure 7.17: Clock recovery and demodulation connected to extra core. Demonstrating a correct response, *ATQA*, to the *REQA* command sent by a NFC reader, here a smartphone of type LG P800.

Part IV

Conclusion and Future Work

Chapter 8

Discussion

In this thesis we have presented an integrated system for a passive NFC tag with integrated sensor front-end and ADC. Handling of the digital NFC-A Tag type 1 protocol is implemented in hardware description language, tested and synthesized into transistor level layout. A power-on reset circuit is implemented to generate a delayed reset of the logic during power up. Modules for sensor front-end, ADC, power harvesting, and NFC communication link are combined with the synthesized core and power-on reset into a single ASIC. A test board is created, and the fabricated chip's digital core and Power-on reset circuit are tested.

Looking back at the goals set at the onset of this thesis, we can look at how well the work presented in this thesis match up with the scope goals.

+ A process for digital design synthesis from HDL to transistor level layout has been created.

+ A NFC-A Tag type 1 protocol digital core, has been created and synthesized using the process mentioned above.

+ Previously made modules handling sensor front-end, ADC, power harvesting and NFC communication link has been combined with a digital protocol core in a singular ASIC.

– The system has not been demonstrated as only needing an external antenna, capacitive energy storage and sensor.

+ A system testing board has been created.

– The test board was not connected to an FPGA board for further testing, due to the testing board's design having level shifters operating on the same higher voltage that the produced ASIC also used for ESD protection, and the ASIC's wrongly oriented ESD protection.

+ A preliminary test of parts of the system have been done using a commercially available NFC reader, here both an LG P800 and a Sony Xperia Z1 have been used to test demodulation and clock recovery, two modules which previously have been tested.

– The complete system is not demonstrated to work solely by harvested power.

– Neither, has it been demonstrated that sensor readout can be successful using just an NFC reader.

Turning our attention towards the objective of this work, to: "Design, implement and test a single passive ASIC NFC sensor tag to enable a small NFC sensing system." A design is presented where a novel NFC-A tag type 1 standard compliant digital design implementation is demonstrated to work with external stimuli modeling the operation of the complete system's internals. Preliminary power consumption figures points at the feasibility of a passive tag operating from an external reader's energy induced in the tag's receiving coil. A tag specification that is initially for data storage read and writes, is adapted to control and readout from an ADC.

8.0.1 Challenges

The error in the produced chip's ESD protection orientation hampered our efforts by effectively shorting the higher input voltage pin to ground when they where above a certain limit. The fix of cutting the ESD protection connections using a focused ion beam worked, but due to small pitch in the pad frame the dies where sent to an external die to package bonder. Unfortunately these bonded chips where not returned to us in time for the completion of this thesis.

8.1 Conclusion

For the overreaching goal of "Demonstrating how a small implantable NFC and sensor tag is feasible for chronic disease monitoring, e.g. diabetes or high blood pressure."

This work does not go into what is needed for the system to be actually implantable nor what specific sensor one would use in such a setting.

The system presented in this work demonstrates how a passive tag protocol can be combined with a complete sensor read-out system, and used to control read-outs from a possible sensor.

The fabricated ASIC have the dimensions of $1.96\text{mm} \times 1.96\text{mm}$, which in turn can be connected with a resistive or capacitive sensor, small capacitive energy storage, and an antenna coil to form a sensory readout system.

8.2 Possible applications

Dangerous things [1] sells an injectable NFC tag. Their NFC chip and antenna coil is encased in a $2 \times 12\text{mm}$ cylindrical biocompatible glass shell, and injected using a syringe, although that solution isn't approved by any safety agency, the basic method used could be.

[1] https://dangerousthings.com/

A working system with sensor and small antenna that is easy to implant and extract again opens a whole range of possibilities. Looking beyond chronically ill, possible users could be athletes looking to track their glucose levels during workout. It could also be used preventively for user at risk of developing a disease, for example diabetes, high blood pressure, or deficiency related diseases. Similar needs could exist in the field of husbandry, tracking how their animals fare, preventing diseases to develop.

8.3 Future work

Testing of ASIC chips where the ESD protection has been cut is the logical next step for this project.

If testing of this system is successful the design would need to be fixed in regards to the ESD protection modules.

To reduce the power draw from the digital module, clock suspension or power gating could be implemented. There is low hanging fruit where power could be saved; e.g. the NFC specification opens up for a receiving module not being active during transmission and visa versa. A central timing circuit could ensure that only the modules that are possible for use, based in NFC protocol specifications, are powered up with full speed clock.

A shorter m3 transistor in POR elements would reduce area use and not change much else of how it preforms.

The front-end and ADC have their own clock recovery and independent rectifier and regulators. This was done to reduce the amount of changes needed to be verified before tape-out. In a future implementation these are obvious improvements that can reduce both area and potential failure points. The rectifier and power regulators could be defended, but the clock recovery is another matter. The two clock recovery circuits could be joined into one.

The integrating ADC requires, as of now, an externally connected VBIAS supply at 0.27V. This bias voltage can be created using an external resistive divider, but in a future implementation one could implement an internal solution for this such that the tag can be even more self contained. For this there are already internal reference generation circuits that could be repurposed.

If a new test board is to be created, separate power planes or separable power planes should be used such that different components one would like to test can be powered individually.

For further development towards an implantable device one would need to implement better over-voltage protection, and implement other rules that health agencies have stipulated.

Bibliography

[1] *Atlys Board Reference Manual, for board Rev C*. Digilent. Aug. 2013.

[2] S.U. Ay. "A nanowatt cascadable delay element for compact power-on-reset (POR) circuits." In: *Circuits and Systems, 2009. MWSCAS '09. 52nd IEEE International Midwest Symposium on*. Aug. 2009, pp. 62–65. DOI: 10.1109/MWSCAS.2009.5236153.

[3] D. Baddeley. *ISO/IEC 14443-2*. Tech. rep. ISO/IEC, Mar. 1999.

[4] Erik Brunvand. "Digital VLSI Chip Design with Cadence and Synopsys CAD Tools." In: 1st ed. Pearson, Mar. 2009. Chap. 11.

[5] A. Dahl, A. Zaher, and T. P. Plagemann. "Android Based Toolset for NFC Tag Testing and Performance Evaluation." In: *European Wireless 2015; 21th European Wireless Conference; Proceedings of*. May 2015, pp. 1–8.

[6] Seventh Edition Committee Diabetes Atlas. *IDF Diabetes Atlas*. 7th ed. International Diabetes Federation, 2015.

[7] Klaus Finkenzeller. *RFID Handbook. Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. 3rd ed. John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom: John Wiley and Sons, 2010.

[8] NFC Forum. *NFC Digital Protocol*. Tech. rep. NFC Forum, Nov. 2010.

[9] NFC Forum. *Type 1 Tag Operation Specification*. Tech. rep. NFC Forum, Apr. 2011.

[10] P. Hafliger and E. Johannessen. "Analog to interval encoder with active use of gate leakage for an implanted blood-sugar sensor." In: *Biomedical Circuits and Systems Conference, 2008. BioCAS 2008. IEEE*. Nov. 2008, pp. 169–172. DOI: 10.1109/BIOCAS.2008.4696901.

[11] K. Kotani, A. Sasaki, and T. Ito. "High-Efficiency Differential-Drive CMOS Rectifier for UHF RFIDs." In: *IEEE Journal of Solid-State Circuits* 44.11 (Nov. 2009), pp. 3011–3018. ISSN: 0018-9200. DOI: 10.1109/JSSC.2009.2028955.

[12] T. T. Nguyen and P. Häfliger. “An energy efficient inverter based readout circuit for capacitive sensor.” In: *Biomedical Circuits and Systems Conference (BioCAS), 2013 IEEE*. Oct. 2013, pp. 326–329. DOI: 10.1109/BioCAS.2013.6679705.

[13] T. T. Nguyen and P. Häfliger. “Inverter based readout circuit for implanted glucose sensor.” In: *Biomedical Circuits and Systems Conference (BioCAS), 2012 IEEE*. Nov. 2012, pp. 252–255. DOI: 10.1109/BioCAS.2012.6418449.

[14] Trung Thanh Nguyen. “Energy efficient implant systems for biomedical applications.” PhD thesis. Faculty of Mathematics and Natural Sciences, University of Oslo, 2015.

[15] Trung Thanh Nguyen, L.A.L. Fernandes, and P. Häfliger. “An Energy-Efficient Implantable Transponder for Biomedical Piezo-Resistance Pressure Sensors.” In: *Sensors Journal, IEEE* 14.6 (June 2014), pp. 1836–1843. ISSN: 1530-437X. DOI: 10.1109/JSEN.2014.2304566.

[16] Trung Thanh Nguyen and P. Hafliger. “A Sub-micro Watt Implantable Capacitive Sensor System for Biomedical Applications.” In: *Circuits and Systems II: Express Briefs, IEEE Transactions on* PP.99 (2014), pp. 1–1. ISSN: 1549-7747. DOI: 10.1109/TCSII.2014.2368260.

[17] INNOVISION RESEARCH and TECHNOLOGY PLC. *Topaz 13.56MHz Near Field Communication (NFC) / Radio Frequency Identification (RFID) Read/Write IC*. Technical datasheet 2. INNOVISION RESEARCH and TECHNOLOGY PLC, June 2007.

[18] N. T. Trung and P. Hafliger. “Time domain ADC for blood glucose implant.” In: *Electronics Letters* 47.26 (Dec. 2011), S18–S20. ISSN: 0013-5194. DOI: 10.1049/el.2011.2685.

[19] Alain Vachoux. *Top-down digital design flow*. http://lsm.epfl.ch/webdav/site/lsm/users/104020/public/Topdown_DF.pdf. Microelectronic Systems Lab, STI-IMM-LSM. Dec. 2005.

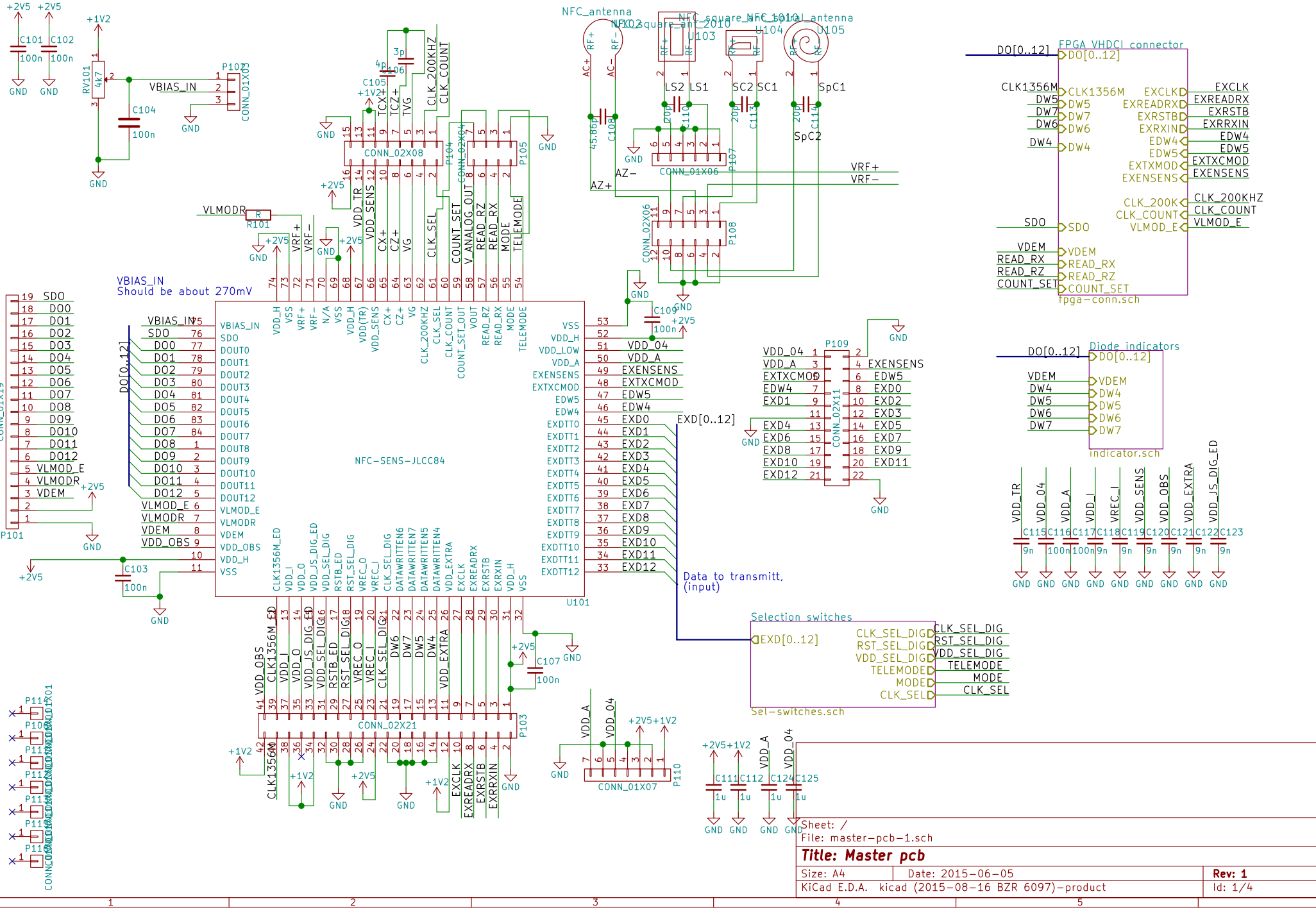
[20] P. Wei et al. “High-Efficiency Differential RF Front-End for a Gen2 RFID Tag.” In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 58.4 (Apr. 2011), pp. 189–194. ISSN: 1549-7747. DOI: 10.1109/TCSII.2011.2124530.

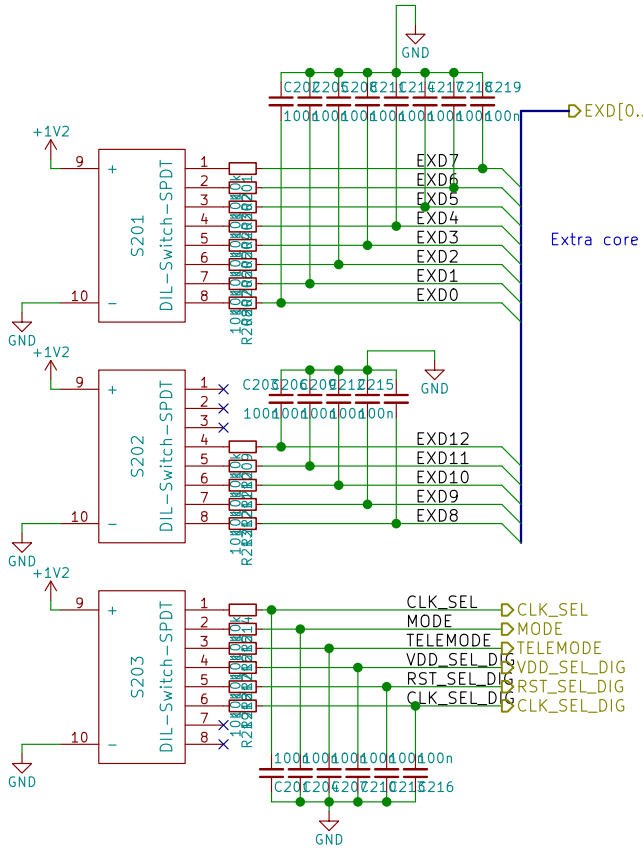
[21] Ali Zaher. “Introducing NFC for in-body and on-body medical sensors.” PhD thesis. Faculty of Mathematics and Natural Sciences, University of Oslo, 2016.

[22] Ali Zaher et al. “Integrated electronic system for implantable sensory NFC tag.” In: *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*. Aug. 2015, pp. 7119–7122. DOI: 10.1109/EMBC.2015.7320033.

Part V

Appendices

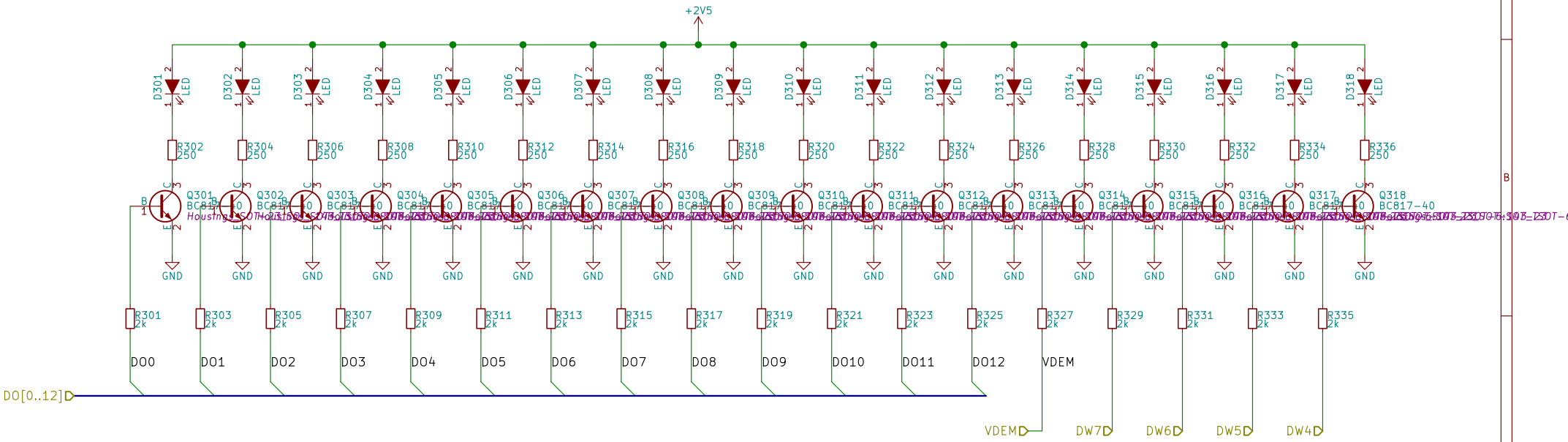

NFC_antenna
NFC square_NFC_1010_antenna
U103
U104
U105
FPGA VHDCI connector
fpga-conn.sch
VBIAS_IN
Should be about 270mV
NFC-SENS-JLCC84
U101
Data to transmit, (input)
Selection switches
Sel-switches.sch
Diode indicators
indicator.sch
Sheet: /
File: master-pcb-1.sch
Title: Master pcb
Size: A4
Date: 2015-06-05
Rev: 1
KiCad E.D.A. kicad (2015-08-16 BZR 6097)-product
Id: 1/4



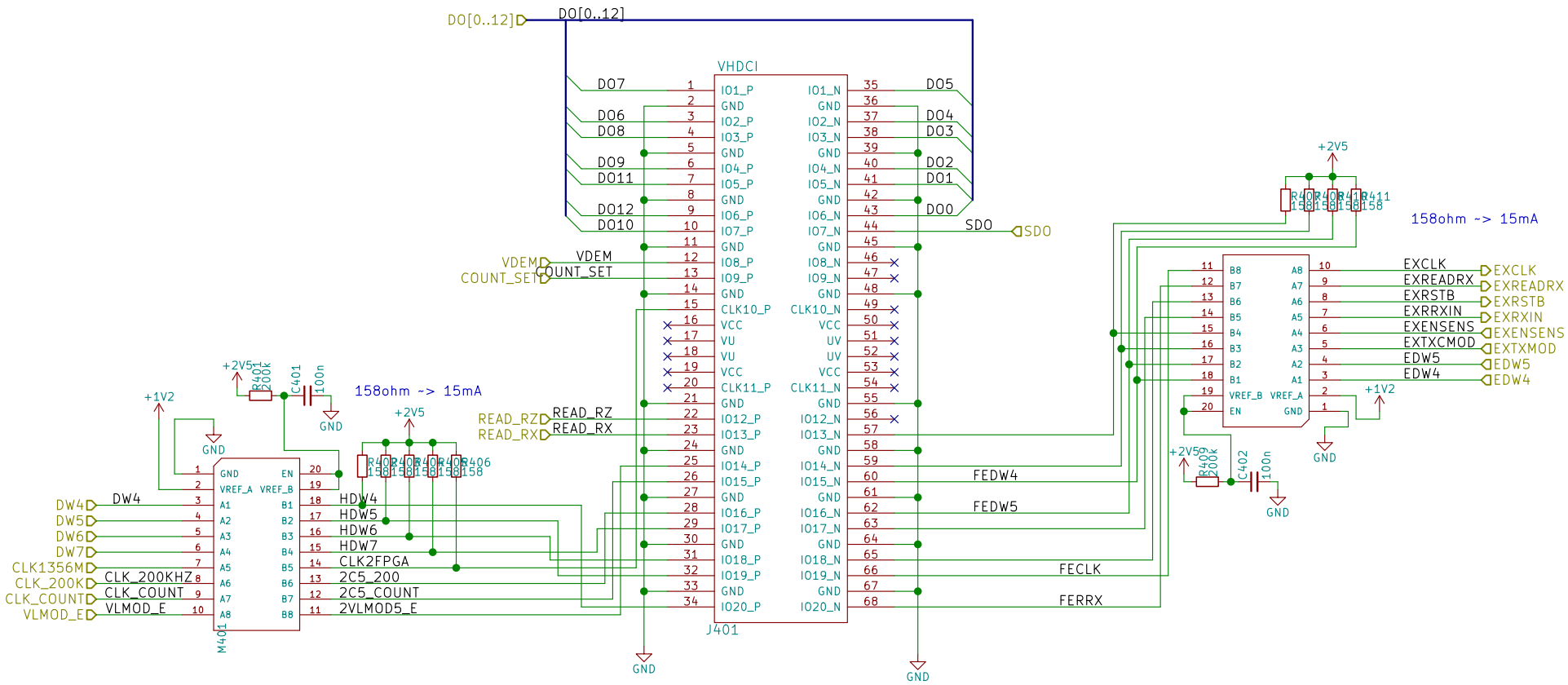
Sheet: /Selection switches/
File: Sel-switches.sch

Title:

Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (2015-08-16 BZR 6097)-product		Id: 2/4



Sheet: /Diode indicators/		
File: indicator.sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (2015-08-16 BZR 6097)-product		Id: 3/4



Sheet: /FPGA VHDCI connector/		
File: fpga-conn.sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (2015-08-16 BZR 6097)-product		Id: 4/4

Synthesis and Place and Route design flow walkthrough

June 16, 2015

1. You need to add the following in your .bashrc file.

```
************
#Cadence Encounter RTL Compiler
RTLCompiler_DIR=/projects/nanos/programs/EncounterRTLCompiler91
PATH=.:$RTLCompiler_DIR/bin:$PATH
export RTLCompiler_DIR PATH
#end Cadence Encounter RTL Compiler

#Cadence Encounter SoC  place and route
SOC_ENCOUNTER_DIR=/projects/nanos/programs/EDI91
PATH=.:$SOC_ENCOUNTER_DIR/bin:$PATH
export SOC_ENCOUNTER_DIR PATH
#end Cadence Encounter SoC  place and route

#Cadence Encounter Conformal equivalence check
CONFORMAL_DIR=/projects/nanos/programsrhel6/CONFRML121
PATH=.:$CONFORMAL_DIR/bin:$PATH
export CONFORMAL_DIR PATH
#end Cadence Encounter Conformal equivalence check

# Licensing server
export LM_LICENSE_FILE=5370@lisens.ifi.uio.no:${LM_LICENSE_FILE}
************
```

   After that, open a new terminal and you'll have access to these tools.

Tool:	Command:
Cadence Encounter RTL Compiler	`rc -gui`
Cadence Encounter Place and route	`encounter`
Cadence Encounter conformal	`lec_auto`

2. Create four folders: one for RTL, one for Simulation, one for Synthesis and finally one for Place and Route.

3. In the synthesis folder, you'll have the `run_synthesis.tcl` script needed to be sourced by cadence RC synthesis tool, as defined in Amirs tutorial. Cadence Encounter RTL Compiler can be started with the command `rc -gui`. Then go to `File -> Source script ...` then a pop-up window titled `Read Tcl Script Files` appear. Browse to `run_synthesis.tcl` file, click on it, and press OK. The important part there is to activate clock gating, and to set the correct libraries. My choice in the end was to use worst case library version of hight Vth cells. This leads to the lowest dynamic and leakage power consumption. Of course the usual cost is a slower design, but in our case 13.56MHz is a very slow design (compared to 800 MHz). Always check the timing reports to see if you have any negative slack.

   If you don't want to run the gui the command `rc -files run_synthesis.tcl` runs synthesis for you. Below is a copy of a `run_synthesis.tcl` file used in a digital design. Notice that all the `set_atter` and other commands should lie on the same line. It has been broken into two lines for the sake of clarity.

```
************
set_attribute hdl_language vhdl
set_attribute information_level 4
set_attribute hdl_latch_keep_feedback true
```

```
set_attribute hdl_error_on_latch false
set_attr lp_insert_clock_gating true
set_attribute lp_power_analysis_effort high

set_attribute hdl_search_path
/ifi/asgard/a03/alizah/work_phd/fpgatools/rtl_folder/rtl_03_no_clk
set_attribute lib_search_path
/projects/nanos/design_kit/TSMC90nmLPRF_616OA_bibl/TSMCHOME/digital/Front_End/timing_power_noise/CCS

##Using HVT with worst-case lib leads to the least power consumption, on the cost of slower design.
set_attribute library {tcbn90lphphvt_150h/tcbn90lphphvtwc_ccs.lib }
set_attribute lef_library
{../../../Back_End/lef/tcbn90lphphvt_150d/lef/tcbn90lphphvt_9lmT2.lef}
set_attribute cap_table_file
../../../Back_End/lef/tcbn90lphphvt_150d/techfiles/cln90_1p09m_top2_rcworst.ict.captable

## Specifies which VHDL files are read by RC
set file_list {rx_tx_control_no_clk.vhd rx_crc_no_clk.vhd rx_decoder_no_clk.vhd
 tx_coding.vhd tx_crc.vhd nfc_top_no_clk_asic.vhd}
read_hdl $file_list

## This buils the general block
elaborate

##Here you define the clocks, clock uncertainties ....
define_clock -name clk1356 -period 60000 [list "clk_1356_g"]
set_attribute clock_source_late_latency 1500 clk1356
set_attribute clock_source_early_latency 1500 clk1356
set_attribute clock_network_late_latency 500 clk1356
set_attribute clock_network_early_latency 500 clk1356
set_attribute clock_setup_uncertainty {1000 1000} clk1356
set_attribute clock_hold_uncertainty 1000  clk1356
define_clock -name clk128 -period 7680000
[find /designs/nfc_top_no_clk/instances_hier/control_a -pin clk_128]
define_clock -name clk64 -period 3840000
[find /designs/nfc_top_no_clk/instances_hier/control_a -pin clk_64]
define_clock -name clk16 -period 960000
[find /designs/nfc_top_no_clk/instances_hier/control_a -pin clk_16]


## Here we define the power optimization. Comment if you don't want to run power optimization
set_attribute max_leakage_power 40000 /designs/nfc_top_no_clk
set_attribute max_dynamic_power 60000 /designs/nfc_top_no_clk
set_attribute lp_power_optimization_weight 0.9 /designs/nfc_top_no_clk

## This synthesizes your code
synthesize -to_mapped -effort high

## This writes all your files in Verilog format
write -mapped > nfc_top.v
write_sdc > nfc_top.sdc

report area > area_nfc_top_asic_hvtpwc.rep
report timing > timing_nfc_top_asic_hvtpwc.rep
report power > power_nfc_top_asic_hvtpwc.rep

##To simulate the verilog netlist in modelsim with timing you need a standard delay format file.
## use the following command to write the timing to file:
write_sdf > nfc_top.sdf
```

```
puts "The RUNTIME is [get_attribute runtime /]s"
************
```

4. Once you finished synthesis, and you are happy with the results, one extra stage needs to be done before you do place and route, Equivalence check. This is to make sure that the netlist generated is functionally equivalent to your RTL. This is done using conformal tool from Cadence. You can start it by writing `lec_auto` in the terminal. Before that, you need to create a do-file from the synthesis tool before you close it using the `write_do_lec` command as shown on this website: `https://lost-contact.mit.edu/afs/ict.kth.se/pkg/cadence/rtls10/01.00.120/doc/rc_user/lec.html` Or using the following command:

   ```
   write_do_lec -revised_design [netlist file] > [output do file]
   ```

   For example,

   ```
   write_do_lec -revised_design nfc_top.v > do_nfc_lec
   ```

   After that, you open conformal(`lec_auto`), select where the log file should be written from the menu "setup" → "log file", and then in the first file menu click on "do do-file" and select the do file you just created. It will run, and in the end you need to make sure that the RTL and the netlist are equivalent. This can be checked in the log file.

   If you want to set up the log file and do file straight from the command line, repurpose this command:

   ```
   lec_auto -Dofile [do file name] -LOGfile [log file name]
   ```

5. Once finished with equivalence check, you can do if you want a netlist simulation, i.e. you run the testbench simulation in Modelsim with the netlist instead of the RTL. To do this, you need to run modelsim. Follow this link to start it `http://robin.wiki.ifi.uio.no/FPGA_tools`. Once finished, you can start modelsim(`vsim`). Then go to "File" → " New" → "project". Choose a name for the project and a proper project location. Click on Ok and then a popup file titled "Add items to the project". Choose "Add existing File" and then browse to choose the netlist file, the testbench file and finally the file comes with the technology file.

   ```
   /projects/nanos/design_kit/TSMC90nmLPRF_616OA_bibl/TSMCHOME/digital/Front_End/verilog/
   tcbn90lphphvt_150j/tcbn90lphphvt.v
   ```

   "Compile " → " Compile all". Once you solve all the errors in compilation, go to "Simulate" → "Start Simulation" ... . A "Start Simulation" pop-up window appears. Under "Design" tab, choose your design under the testbench name. Unclick "enable optimization", and then under SDF tab, add the sdf file we generated in synthesis process. Choose typical as an option. Then click Ok, and from there you can run your simulations.

6. Last tool to use is Cadence encounter Place and route (started with command: `encounter`). The following is a retelling but somewhat changed procedure from Amirs tutorial.

   (a) First, import your synthesised design. "design" → "Import Design". In the Basic tab:
   - Verilog Netlist: [.v] Created using Encounter RTL compiler
   - Timing Libraries: [.lib] Created using Encounter Library Characterizer
   - LEF files: [.lef] Generated by Virtuoso LEF Out
   - Timing Constraints: [.sdc] Generated by Encounter RTL compiler
   - IO Assignment files: [.io] (Optional)

   Some of the files may be found by inspecting the ".tcl" file used in the synthesis. The files i used where:

base dir:	`/projects/nanos/design_kit/TSMC90nmLPRF_616OA_bibl/TSMCHOME/digital/`
lef file:	base dir + `Back_End/lef/tcbn90lphphvt_150d/lef/tcbn90lphphvt_9lmT2.lef`
timing lib:	base dir + `Front_End/timing_power_noise/CCS/tcbn90lphphvt_150h/tcbn90lphphvtwc_ccs.lib`
timing constraint:	`../synthesis/nfc_top.sdc`

   Under the advanced tab → power, specify "vdd!" and "gnd!" for power and ground nets.

   And in "RC Extraction" set the capacitance table files. The ones I used where under the path:
   `/projects/nanos/design_kit/TSMC90nmLPRF_616OA_bibl/TSMCHOME/digital/Back_End/lef/tcbn90lphphvt_150d/techfiles/` with the filenames:

   typical `cln90_1p09m_top2_typical.ict.captable`
   best `cln90_1p09m_top2_rcbest.ict.captable`
   worst `cln90_1p09m_top2_rcworst.ict.captable`

   When you are done with this setup you can save it as a conf file by clicking the "save" button next to the "ok" button, the conf file can in turn be loaded with encounter next time you need it with the command:

```
encounter -config <name of config file>
```

(b) Next you need to specify the sice of your floorplan, this is done under"Floorplan" → "Specify Floorplan...". One thing to take into consideration are guard rings, if you are planning to implement any, they need some space outside the core area that's inside the outer bounderies.

(c) Assign positioning and signaltypes for pins under "Edit" → "Pin Editor" .

(d) In "Power" → "Connect Global Nets" add VDD pin, select Apply all under scope, and use use "vdd!" under to global net. Do the same for VSS, except use "gnd!" global net. You could add another connection for VDD and VSS where instead of pin you select Tie High and Tie Low respectively. From the verilog netlist we also need to add connections for `1'b1` and `1'b0` to "vdd!" and "gnd!", instead of pin select Tie High and Tie Low respectively.

Click apply and check, and see if there is any error messages in the terminal output.

(e) Add power rings under "Power" → "Power Planning" → "Add ring". Set width to 0.42 , spacing to 0.5, set offset to center in channel.

You could also add some power stripes from the same menu, but I will not go into that at this time.

(f) Placing of the design, go to "Place" → "Standard cell..." and click ok.

(g) Add the decoupling capacitor cellnames and capacitance(femto farad) into a file called `decap.cells` Here as an example I have used arbitrary values for each decap cell.

```
DCAPHVT 1
DCAPHVT4 4
DCAPHVT8 8
DCAPHVT16 16
DCAPHVT32 32
DCAPHVT64 64
```

Then in encounter command promt use the command

`adddecapcellcandidates -fromFile decap.cells` to define the cells as decap. Then using the command

`addDeCap-totCap[totalcapacitanceinfemtofarad]-cellsDCAPHVTDCAPHVT4DCAPHVT8DCAPHVT16DCAPHVT32DCAPHV`
encounter will place out decaps cells to match your specified amount of capacitance.

(h) To generate clock tree go to "Clock" → "Synthesize Clock Tree" in the menu bar. If you don't have any specs set up, click "gen spec.." and add all inverter and buffer cells from you linked library, then click "ok". The clock drivers(buffers) are named with the prefix CKB and CKBX, while inverting clock drivers are named with the prefix CKN and CKNX.

(i) Add fillers. "Place" → "Physical Cell" → "Add Filler". Add your filler cells using the "select" button next to "cell names", click "ok" and watch all empty spots be filled in your layout. Note that "FILL_NW..." and "FILLHVT1_LL" are fill cells for level shift cells, so if you only have one power domain, don't use them.

(j) Route power and ground nets. "Route" → "Special route". Click ok.

(k) Final routing: "Route" → "NanoRoute" → "Route". Leave everything default and click ok. Lean back and bask in your creation.

7. You may want to add matal fill before finishing.

8. Save design as verilog netlist, for simulation in ModelSim. In the menu click "File" → "Save" → "netlist", and press ok to save.

9. to save a standard delay format file of your design's timing, for further use in modelsim, choose "Timing" → "Write SDF"

10. Transferring design into Cadence Virtuoso.

To do that you need the following 3 steps:

(a) create a new library which should be 'reference' the technology library (tsmcN90rf)

(b) import the LEF with technology info on this new library, it should create all missing vias and layers

(c) save your design from Encounter in this new library. This can be done by saving encounter design as DEF, and then importing this DEF to the new library.

Now into details about each step:
******

First step: create a new library which should be 'reference' the technology library (tsmcN90rf)
******

(a) In Library Manager in Virtuoso, go to "File" → "New" →"Library".

(b) In the field "Name", enter the library name: `"Digital_Design_Tutorial"`. Press Ok.

(c) A pop-up window Technology File for New Library appears. Choose "Reference existing technology libraries". Click Ok.

(d) In the next window, choose "tsmcN90rf" from Technology Libraries menu, click on → to added it to Reference Technology Libraries. Click on OK.

(e) Now, the new empty library is created.

******

Second step: import the LEF with technology info on this new library
******

(a) First locate your LEF file you are using with Encounter. In this case it is: `/projects/nanos/design_kit/TSMC90nmLPRF_616OA_bibl/TSMCHOME/digital/Back_End/lef/tcbn90lphphvt_150d/lef/tcbn90lphphvt_9lmT2.lef`

(b) Go to Virtuoso window, "File"→"Import"→ "LEF.."

(c) Virtuoso(R) LEF window appears.

(d) In "LEF file name" enter `/projects/nanos/design_kit/TSMC90nmLPRF_616OA_bibl/TSMCHOME/digital/Back_End/lef/tcbn90lphphvt_150d/lef/tcbn90lphphvt_9lmT2.lef`

(e) In "Target Library name" , enter `Digital_Design_Tutorial`.

(f) All the rest fields are empty.(except Use GUI fields). Press OK.

(g) It will take some time adding the LEF files. Once finished, you'll get a message about lefin translation is finished with so many errors and warning. Check lefin.log file in the directory to see what errors are there.

(h) Now, you can see that Digital Design Tutorial Library is no longer empty. It has all the abstract views of all the cells in tcbn90lphphvt.

******

Third step: Save and import DEF file from Encounter to Virtuoso
******

(a) In encounter terminal, run the following command: `defOut -floorplan -netlist -routing -unit 1000 encounter`

(b) Back in Virtuoso, go to "file"→ "Import"→ "DEF.."

(c) in "DEFIn File Name", add the path to `encounter_design_name.def`.

(d) in "Target library name", add `Digital_Design_Tutorial`.

(e) Ref. Technology should be empty.

(f) Fill in "Target Cell name", with the name of the top entity in your digital design as it appears in Encounter.

(g) Fill in "target view name" with layout..Click Ok.

(h) Once finished, a popup message will show how many errors and warning. Check defin.log file in your directory.

(i) Now you have the layout of your cell in `Digital_Design_Tutorial` library and you can use it by adding it as an instance in any layout.

******

Extra step: Export GDS file in Virtuoso
******

(a) Back in Virtuoso, go to "File"→ "Export"→ " Stream.."

(b) In Stream file, enter the name of the gds file: `digital_design_stream.gds.gz`

(c) Leave "technology library" field empty.

(d) IN "Library" field, add `Digital_Design_Tutorial`

(e) Add the top Level Cell.

(f) Leave the rest as they are.(View: layout)

(g) Click on Translate.

(h) Once finished, a popup message will show how many errors and warning. Check streamIn.log file in your directory.

Parts this walkthrough does not address

(a) Double checking timing after place and route is complete.

(b) Optimizing for power

You may need a good tutorial to follow there, and I can recommend one: `http://www.cs.utah.edu/~elb/cadbook/color-figs/Chapter11-Encounter.pdf`

Table 8.1: BOM, ordered from Farnell, part 1 of 2

Line Num.	Farnell Part Number	To Order	Cost p/unit (NOK)	Total Cost	Name Of Component	Other
1	1685056	18	1.28	23.04	ROHM SML-211UTT86K LED, 0805, RED, 2.5MCD, 620NM	orange led
2	2432140	2	17.2	34.4	TEXAS INSTRUMENTS LSF0108PWR VOLT LEVEL TRANSLATOR, OCTAL, TSSOP-20	Levelshifters
3	1798081	18	0.178	3.204	MULTICOMP BC817-40 TRANSISTOR, NPN, 0.5A, 45V, SOT23	transistors
4	1522015	3	8.5	25.5	MULTICOMP MCTIMR-08 SWITCH, DIL, TRI-STATE, SMD, 8WAY	switches
5	1667525	1	18.14	18.14	SAMTEC CES-120-01-T-S RECEPTACLE, 2.54MM, SINGLE, 20WAY	receptacles 1x20
6	1593495	1	8.2	8.2	MULTICOMP 2214S-22SG-85 SOCKET, PCB, 2 ROW, VERT, 22WAY	receptacles 2X11
7	1593489	1	3.04	3.04	MULTICOMP 2214S-08SG-85 SOCKET, PCB, 2 ROW, VERT, 8WAY	receptacles 2X4
8	2356149	1	18.45	18.45	WURTH ELEKTRONIK 61304421121 HEADER, 2.54MM, PIN, THT, VERT, 44WAY	pins 2x22
9	1593445	1	1.1	1.1	MULTICOMP 2213S-16G HEADER, 2 ROW, VERT, 16WAY	pins 2X8
10	1022238	1	2.42	2.42	HARWIN M20-9980646 HEADER, 2ROW, 12WAY	pins 2X6
11	2356166	1	8.8	8.8	WURTH ELEKTRONIK 61301611121 HEADER, 2.54MM, PIN, THT, VERT, 16WAY	pins 1X9
12	2320936	29	0.68	19.72	MULTICOMP MC1210B104K500CT SMD Multilayer Ceramic Capacitor, MC Series, 0.1 µF, ± 10%, X7R, 50 V, 1210 [3225 Metric]	100nF
13	2332901	4	1.6	6.4	AVX 12101C105MAT2A SMD Multilayer Ceramic Capacitor, 1 µF, ± 20%, X7R, 100 V, 1210 [3225 Metric]	1uF
14	2320946	7	1.4	9.8	MULTICOMP MC1210B103K501CT SMD Multilayer Ceramic Capacitor, MC Series, 0.01 µF, ± 10%, X7R, 500 V, 1210 [3225 Metric]	10nF

Table 8.2: BOM, ordered from Farnell, part 2 of 2

Line Num.	Farnell Part Number	To Order	Cost p/unit (NOK)	Total Cost	Name Of Component	Other
15	1470031	21	0.763	16.023	VISHAY DRALORIC CRCW121010K0FKEA Surface Mount Chip Resistor, Thick Film, AEC-Q200 CRCW Series, 10 kohm, 330 mW, ± 1%, 200 V	10k
16	2312516	19	1.61	30.59	PANASONIC ELECTRONIC COMPONENTS ERJP14F2051U Surface Mount Chip Resistor, Thick Film, AEC-Q200 ERJ Series, 2.05 kohm, 500 mW, ± 1%, 200 V	2k
17	2312548	19	1.58	30.02	PANASONIC ELECTRONIC COMPONENTS ERJP14F2490U Surface Mount Chip Resistor, Thick Film, AEC-Q200 ERJ Series, 249 ohm, 500 mW, ± 1%, 200 V	250R
18	2380714	3	1.05	3.15	PANASONIC ELECTRONIC COMPONENTS ERJU14F2003U Surface Mount Chip Resistor, Thick Film, AEC-Q200 ERJ Series, 200 kohm, 500 mW, ± 1%, 200 V	200k
19	2312474	10	1.58	15.8	PANASONIC ELECTRONIC COMPONENTS ERJP14F1580U Surface Mount Chip Resistor, Thick Film, AEC-Q200 ERJ Series, 158 ohm, 500 mW, ± 1%, 200 V	158R
20	1771725	5	1.62	8.1	MURATA PVZ2A103C04B00 TRIMMER, SMD, 10K	pot 10k
21	1961257	4	3.9	15.6	VISHAY VITRAMON VJ1210A101JXRAT5Z SMD Multilayer Ceramic Capacitor, Arc Guard®, VJ HV Series, 100 pF, ± 5%, C0G / NP0, 1.5 kV	100pF
22	1856524	30	0.101	3.03	MCSH31B221K160CT	220pF
23	1855856	30	0.318	9.54	MC1206N680J202CT	68pF
24	1855796	30	0.318	9.54	MC1206N470J201CT	47pF
25	1855816	30	0.348	10.44	MC1206N220J631CT	22pF
26	1856537RL	60	0.0979	5.874	MCSH31B471K160CT	470pF
27	2320889	3	0.402	1.206	MC1206N100J500CT	10pF
28	1855829	30	0.286	8.58	MC1206N3R3C102CT	3.3pF
29	1855808	30	0.216	6.48	MC1206N1R5C631CT	1.5pF
30	N/A	1	90	90	VHDCI connector [2]	