CHALMERS

Profibus PA interface for the HMS Anybus

Master's Thesis in Embedded Electronic System Design

Jacob Rosén

Chalmers University of Technology
Department Computer science engineering
Gothenburg, Sweden, June 2015

A Profibus PA interface for the HMS Anybus

JACOB ROSÉN

Examiner: PER LARSSON-EDEFORS

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: The cover shows a rendering of a ABCC40-Profibus PA module

Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Abstract

There are a large number of competing standards for industrial communication, depending on location and market: when developing a product one has to choose what interface to use. A solution is to use a simpler interface provided by a third party, such as the Anybus by HMS Networks to connect the network of choice. Using the Anybus means only one interface has to be developed. An interface between Profibus PA and Anybus has been created to expand the available connectivity for the Anybus. The interface has successfully been implemented in the ABCC form factor using the NP40 SoC.

Acknowledgements

First I would like to thank my advisors at HMS, Peter Anderson and Henrik Erlund for helping me to start and finish the project respectively. I would also like to thank everyone at the hardware department at HMS for helping me with utilizing equipment and answering questions. I would like to thank Torbjörn Ferdman for answering all and any questions about components and errors during the design of the PCB, and Jörgen Pålsson for indispensable help with the HDL and the development tools used to synthesize the code. I would also like to thank Fedor Meyer from Procentec for his help with Profibus PA related questions. Finally I want to thank my supervisor Sven Knutson for his feedback and thoughts on the report, as well as my examiner Per Larsson-Edefors for his feedback and detailed comments on my report.

Jacob Rosén, Halmstad June 17, 2015

Contents

Acronyms

ABCC Anybus CompactCom

AHB Advanced High-Performance Bus

APB Advanced Peripheral Bus

CAN Controller Area Network

CAT 6 Category 6 ethernet cable

CRC Cyclic Redundancy Check

DPV1 Decentralized Peripherals Version 1

eNVM Embedded Non-Volatile Memory

EOF End Of Frame

eSRAM Embedded Static Random-Access Memory

FIFO First Input First Output

FMS Field Messaging System

FPGA Field Programmable Gate Array

HART Highway Addressable Remote Transducer

HD Hamming Distance

HDL Hardware Description Language

IC Integrated Circuit

IEC International Electrotechnical Commission

IP Intellectual Property

JTAG Joint Test Action Group

LFSR Linear Feedback Shift Register

LUT Lookup Table

NP40 Network Processor 40

OP amplifier Operation Amplifier

OSI Open Systems Interconnection

PC Personal Computer

PCB Printed Circuit Board

PCI Peripheral Component Interconnect

PLC Programmable Logic Controller

Profibus Process Fieldbus

Profibus DP Process Fieldbus Decentralized Peripherals

Profibus PA Process Fieldbus Process Automation

PTC Positive Temperature Coefficient

RAM Random-Access Memory

RTS Request to send

RXA Receiver Activity

RXD Receiver Data

SoC System on Chip

SOF Start Of Frame

SPI Serial Peripheral Interface

SRAM Static Random-Access Memory

TXD Transmitter Data

UART Universal Asynchronous Receiver/Transmitter

USB Universal Serial Bus

1 Introduction

This chapter covers the base of this thesis. Initially the context will be explained, followed by the background. Then the purpose and aim will be covered in section 1.3. After this the scope will be defined in section 1.4. Finally the thesis outline is presented in section 1.5.

1.1 Context

To communicate between machines in the industry some kind of interface is needed. The requirements for this interface differ between applications, e.g. throughput speed, distance between nodes, and if the bus also supplies power. The biggest differences between these fieldbuses and usual computer-buses such as Peripheral Component Interconnect (PCI), Universal Serial Bus (USB), or Firewire are that industry buses usually cover larger distances at the cost of reduced throughput and are often more robust.

Multiple fieldbus standards have been created to meet the requirements of the industry. Some examples of fieldbus standards are: Modbus, Process Fieldbus (Profibus), EtherCAT and Controller Area Network (CAN). These standards can share bus properties, such as the physical transport medium. Different types of industry applications use different fieldbuses due to different needs. Consider the two cases:

1. Communication between assembling robots
2. Communication with level sensors at an oil refinery

In the first case, high throughput is required to minimize waiting time between operations. In contrast, the cables will not have to be long as the distance between the robots will be in the range of meters.

In the second case, the distance between the level sensors can be in the range of kilometers. There is also an explosion hazard, so using the same cables for power and communication are preferred as this both reduces cost and improves safety. Since the level is not likely to change significantly in the time span of seconds, the throughput speed is not a main priority.

As can be seen in these two cases, there are different needs for different situations and different kinds of buses cover these needs. In addition to the difference in performance there are also several fieldbuses with similar performance. The bus used by a device can depend on what geographical area the target market is, and what fieldbus is present in other devices that are being produced and used. Different fieldbuses cause problems for developers as they have to limit a product's market area depending on what interface they have chosen. A solution for this problem is to use the Anybus [1] developed by HMS Networks which interfaces a sensor, actuator or other device with a fieldbus. Using the Anybus makes it possible to design a device with just one interface and then just plug in the module for the fieldbus of choice.

1.2 Background

As mentioned in Section 1.1, some kind of communication is needed to communicate with sensors and actuators in an industry process. A simple way of doing this is to read the voltage level from a sensor, such as a Positive Temperature Coefficient (PTC) resistor, or to control an actuator by increasing the voltage given to it. However, long cables will have a voltage drop, resulting in incorrect values. One method for solving this issue is to use a current loop. The current output indicates the value, the signal usually ranges from 4 mA to 20 mA, Figure 1 shows a sample current loop. As the current is unaffected by long wires, sensors can be placed far from the unit processing the data. Using this simple interface has the limit of only reporting raw sensor input values, as well as needing separate cabling for each sensor and actuator [2].

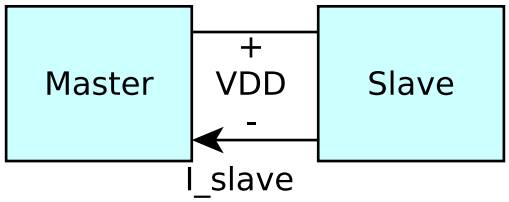

Figure 1: A current loop to read a sensor or control an actuator.

To increase the utility, instead of just sending the raw sensor value, the Highway Addressable Remote Transducer (HART) bus [3] can be used. HART builds on the same physical medium, but sends digital signals where a logical 0 is when the current is less than 3.8 mA and a logical 1 is when the current is above 20.5 mA. Using HART, diagnostic data can be read or written to the sensor. One of the drawbacks of the HART bus is the low baud rate of only 1.2 kbaud [2, 4].

Process Fieldbus Process Automation (Profibus PA) [5, 6] is a fieldbus developed as a competitor and successor to HART systems. As the name suggests, the target application for this bus is in the process industry and the properties of the bus reflect this. The bus uses a two wire interface, with both signal and power supplies in the same differential pair. The data are transmitted at a rate of 31.25 kbaud, this low speed is chosen to allow for long cables. The limiting factor of the cable length is determined by when the signal is dampened more than 6 dB. For longer cables the rise and fall time will be longer, and if the rise and fall time exceeds half of the pulse time there is a high risk of misreading the signal. The maximal length possible is as a function of bit rate shown in Figure 2 taken from the ITU-T Recommendation V.11 [7]. Using termination at the end of the bus increases the performance as it reduces the reflections in the cable. The termination usually consists of a resistor at the end of the bus [7].

The rise time is the transition time it takes for the transition from a logical 0 to a logical 1 and the fall time is the time it takes for a transition a logical 1 to a logical

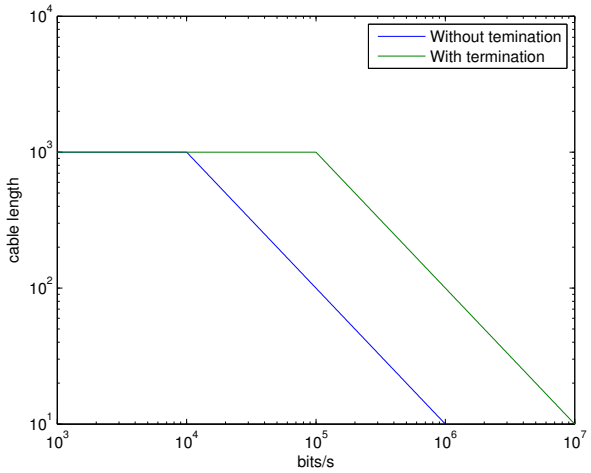

Figure 2: Maximal length for a cable with a diameter of 0.51 mm as a function of bitrate. A termination of 100Ω can be connected to improve performance.

0. The difference between the ideal and actual transitions are shown in Figure 3. The behavior of the actual curve can be described as an exponential, as a cable will induce capacitance creating a lowpass filter. Due to the filtering of the signal, it will take longer time to reach V_{CC}, so the rise and fall time are instead defined as the time between 0.1 V_{CC} and 0.9 V_{CC}. The pulse time is the width of the pulse, it is defined as the time the pulse is above 50% [7].

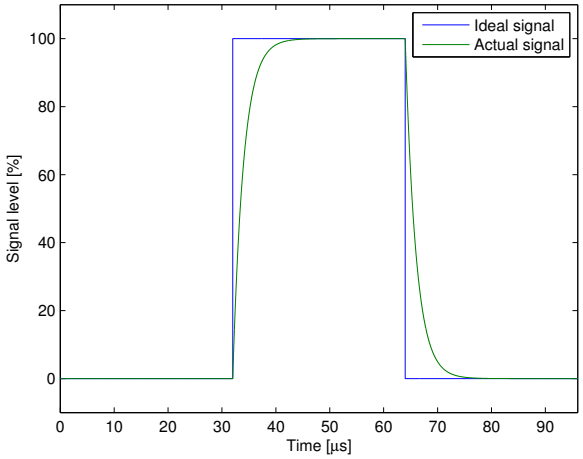

Figure 3: An ideal pulse, and an example of an actual pulse with a frequency of 31.25 kHz. The rise and fall time of the ideal pulse are 0, while the rise and fall time for the actual are 4.39 μs. The pulse width of both the signals is 32 μs.

Due to the low data rate in Profibus PA and special cabling, the cable lengths can be up to 1900 m without repeaters, or up to 9600 m using repeaters [5]. These cable lengths can be compared to EtherCAT using a Category 6 ethernet cable (CAT 6) which has a maximal length of 100 m [6, 8]. The possibility to connect a device from this bus to a standardized more general interface, such as the HMS Anybus, will extend the usability and connectivity of the bus further. There are over 40 million Profibus devices (2012) active in the process industry and over 9 million active Profibus PA devices (2014) [9, 10].

1.3 Purpose / Aim

To increase their product range HMS strives to cover more fieldbuses. One interesting fieldbus is the Profibus PA used in the process industry. The Profibus PA is interesting due to the amount of already present devices and its similarities to the already supported Process Fieldbus Decentralized Peripherals (Profibus DP). The purpose of this thesis is to create an Anybus CompactCom (ABCC)-40 [1] Profibus PA slave module. The main topic of the thesis is the creation and implementation of a bridge between a Decentralized Peripherals Version 1 (DPV1) [11] implementation and the two wires in the Profibus PA.

In case of extra time it would be preferable to adapt the concept module to a prototype on the ABCC form factor.

The Profibus PA implementation will be done in the Field Programmable Gate Array (FPGA) part of one of HMS own chips, the Network Processor 40 (NP40). Additionally, the interfacing will make use of the off-the-shelf Integrated Circuit (IC) AMIS-492x0 [12] to convert the FPGA logic levels to the Profibus PA levels. The AMIS-chip will also need discrete components such as transistors, diodes and passives to complete the level transformation.

1.4 Scope

During the thesis some limitations were set up. These are motivated by the fact that the module developed is only a proof of concept. The limitations were:

- The possibility of using another System on Chip (SoC) was not considered, as this would require software to be rewritten. Being forced to use the same SoC also puts a limitation on how large the Hardware Description Language (HDL) part of the project can be, as there is no possibility to choose a SoC with additional programmable logic. This limitation also means that the HDL can be written specifically for this SoC.
- Power consumption was not considered.
- Financial aspects such as different numbers of layers of the Printed Circuit Board (PCB) or different tolerances on components were not considered.
- Verification was limited to not include test benches for each component. The module was instead validated by using a Profibus PA network and sending data over said bus.

- Delay through the implementation was not considered.
- Lost or corrupt frames were allowed.
- The module was not designed for use in explosive areas.
- The module's performance will only be focused on working for Profibus PA.
- The form factor was allowed to deviate from the ABCC specification, but a solution following the specification was preferred.

1.5 Thesis outline

This thesis starts with Chapter 2 presenting already present technologies that were used to create the module. Chapter 3 describes what was needed to complete the ABCC module, which includes both hardware and software, and includes how the development and verification of the module were done. Chapter 4 describes how the different parts of the module were implemented. Finally the result of the thesis is presented in Chapter 5, a discussion is given in Chapter 6 and conclusions are drawn in Chapter 7.

2 Existing Technology

This chapter will cover the present technology utilized for the project. First the HMS Anybus will be described briefly in Section 2.1, followed by a review of Manchester encoding in Section 2.2. The principle of Cyclic Redundancy Check (CRC) is presented in Section 2.3, followed by an introduction of the Profibus in Section 2.4. Finally the functionality of the AMIS-492x0 is covered in Section 2.5.

2.1 Anybus

Anybus is HMS's interface towards a sensor or actuator. Figure 4 displays the general dataflow of an ABCC-40 and is taken from Hardware Design Guide [13]. The Anybus interface is located at the left side, the Anybus CPU in the middle handles the communication between the Anybus and the Communication Controller. The network interface is located on the right side of the figure, this interface is different depending on the what fieldbus it interfaces.

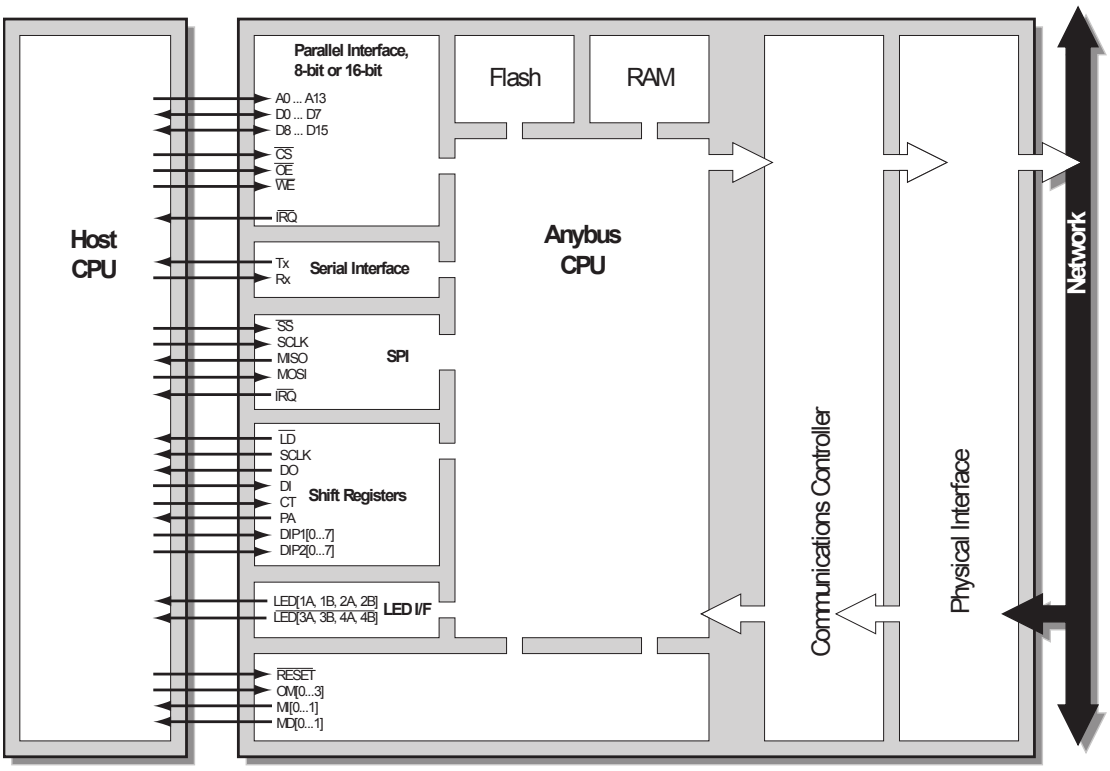

Figure 4: Overview of the ABCC-40 chip

This interface supports different interfaces to ease development towards the Anybus. The available interfaces are: 8/16 bit parallel, serial interface, Shift register, and Serial Peripheral Interface (SPI). The 8/16 bit parallel works like a Static Random-Access

Memory (SRAM) with a 14 bit address. The word length can either be 8 or 16 bits. The serial interface uses Universal Asynchronous Receiver/Transmitter (UART) as its physical medium, supporting speeds from 19.2 kbaud up to 625 kbaud. The SPI interface acts as a SPI slave and supports transmissions in full duplex with a maximal frequency of 25 MHz. Finally, the shift register can be used if the Anybus should act without any external master, shifting data out serially. Only one of these interfaces can be used at once, and the active interface can only be changed by resetting the circuit [13].

2.1.1 ABCC

The ABCC-40 [1] is the newest Anybus version, being the successor of the ABCC-30 [14]. ABCC-40 uses a custom made SoC called NP40 [15], which is based on the Microsemi Smartfusion 2 [16]. The SmartFusion 2 is marketed as a SoC with low power consumption and high reliability. The SoC contains an embedded ARM Cortex M3 processor, an Embedded Non-Volatile Memory (eNVM), an Embedded Static Random-Access Memory (eSRAM), controllers for several different peripherals, and a non-volatile flash-based FPGA fabric. Communication between the CPU and FPGA fabric is done through Advanced High-Performance Bus (AHB)-lite [17] or Advanced Peripheral Bus (APB) 3 [18]. The FPGA fabric is composed of logic elements with a 4 input Lookup Table (LUT) and a D-flipflop. A carry between two bordering logic elements allows for the creation of high-speed adders. The FPGA fabric also contains Math Blocks and fabric Random-Access Memory (RAM) [15, 16, 19, 20].

The ABCC is a standardized physical design developed by HMS, whose module design is shown in Figure 5. The ABCC form factor contains two sides: the Anybus side, on the left, and a fieldbus side, on the right. These two sides are galvanically isolated and are separated by an 2.5 mm air-gap. There are other ABCC form factors available, but those are outside the scope of this thesis.

2.2 Manchester code

Manchester code was first used on the magnetic storage device of the Manchester university mark 1 computer [21], and is a method to serially send data and the clock over one channel at the same time. When using Manchester encoding the signal always has an edge in the middle of the bit. The encoding is done by actually using two bits, encoding a 0 as 01 and a 1 as 10. In addition to 1 and 0, there is also the possibility to send N+ and N-. These two extra characters do not contain an edge and are rarely used. N+ will be encoded as 11 and N- as 00. These special characters can be used to indicate a start or end of frame. Compared to using the raw data bits, Manchester encoding has multiple benefits due to there being at least one transition for each bit:

- There is no need to send the clock as this is superimposed on the data in the same wire.
- There will be no long sections of the signal being the same value. Regularly, long sequences can cause the decoder to lose synchronization.

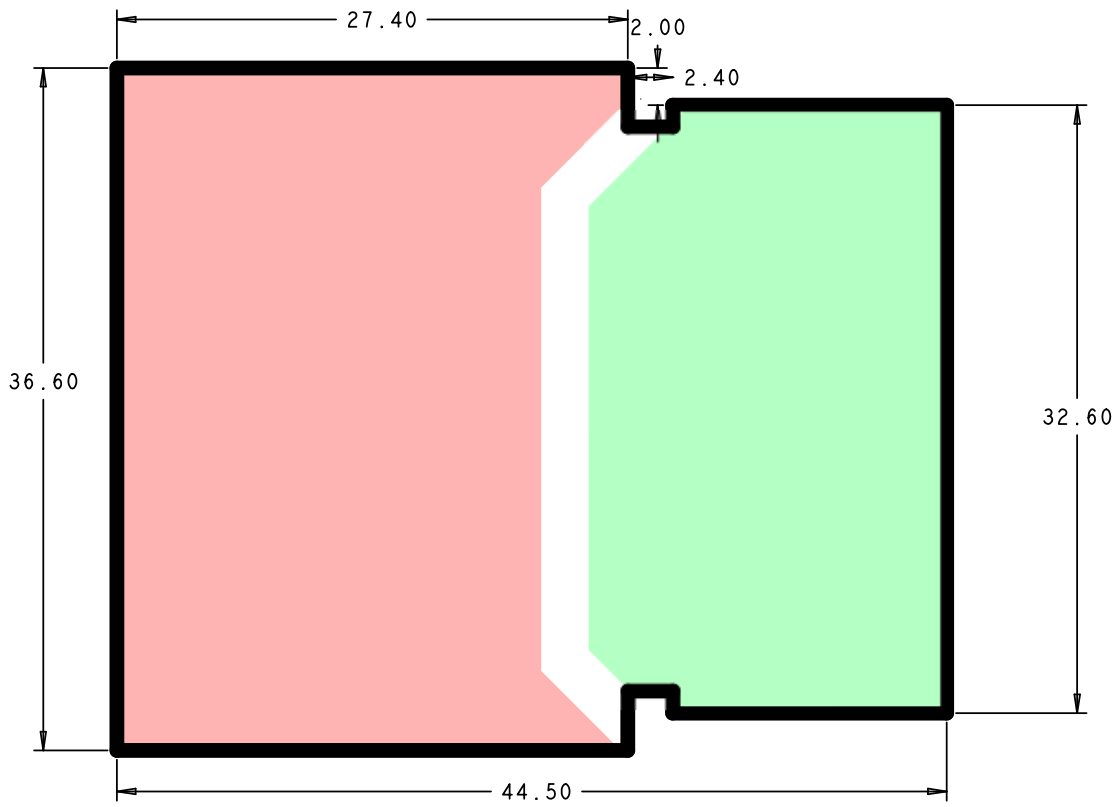

Figure 5: The physical dimensions of the ABCC PCB, all measurements are in mm. The Anybus connector and NP40 is on the left side (red),the fieldbus interface is placed on the right side (green). These sides are divided by a 2.5 mm air gap as the voltages may differ.

- The signal does not contain any information at low frequencies, making it possible to transmit it over a DC line.

Compared to uncoded transmission the Manchester encoded data requires twice the bandwidth.

To encode a bit, the bit and the clock that drives it are combined by a logical exclusive OR (XOR), an example of this is displayed in Figure 6. In this example the output is also inverted as this is the case for Profibus PA [22, 23].

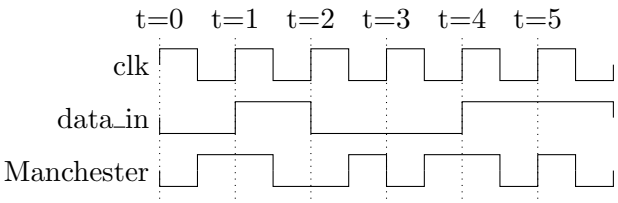

Figure 6: Inverted Manchester encoding of example data

2.3 Cyclic redundancy check

When transmitting data the integrity of it needs to be verified. This verification is usually done by appending a redundant part at the end of the message. The performance of redundant codes can be measured in Hamming Distance (HD). The HD describes how many wrongful bits it takes to misinterpret the message as correct. The HD only tells the worst case, i.e., if a code can detect 5 errors in all cases but one, where it can only detect 3 errors, has a HD = 3.

A common way to implement error detection is to use a checksum: all sent words are added and then the sum is sent at the end of the message. This primitive checksum can be easily calculated at the cost of it being prone to errors as it has a HD = 2. The checksum is efficient to calculate in a processor as it only requires an add operation.

Another method is to use a CRC. The CRC uses the rest of a division as the redundant data. Depending on the length of the CRC and the divisor different HDs can be achieved, but the HD is also dependent on the length of the message: longer messages yields a lower HD. A CRC will always be able to detect one error regardless of divisor or length. Calculating CRC in a processor is relatively expensive as several operations have to be done. If programmable logic instead is used the CRC can be calculated using a Linear Feedback Shift Register (LFSR). The LFSR is a more efficient implementation than to implement the adder needed for a checksum calculation [24].

The LFSR calculates the CRCs bitwise and consists of XOR-gates and registers. The number of registers needed are the same as the order of the CRC. Each clock cycle the values in the registers are shifted, and possibly XORed with the input and output from the previous shift register. The input bits are XORed with the output of the last shift register, the result of this calculation is then XORed into the registers specified by the generator polynomial and the first register. After the last bit is received the CRC is the values in the registers. [25].

2.4 Profibus

Profibus is one of ten fieldbuses standardized by the Industrial communication networks standard International Electrotechnical Commission (IEC) 61158 [6]. The Profibus contains two main sub-standards: the DP and PA, which will be explained in the following subsections. Additionally, there is a third standard called Field Messaging System (FMS) used for master-to-master communications. Profibus DP and Profibus PA were the relevant standards for this thesis. [26].

The Profibus includes three Open Systems Interconnection (OSI) layers: the application layer, the data link layer and the physical layers, as shown in Figure 8. The application layer and data link layer are shared between Profibus DP and Profibus PA[5, 6], but the physical layers differ, Profibus PA also have an addition of the data link layer. The physical layer of the different standards are: Profibus DP has different throughput rates ranging from 9600 kbaud to 12 Mbaud, while Profibus PA only has one throughput rate of 31.25 kbaud. The Profibus PA also has the possibility to power the slave device through the two data lines, while Profibus DP uses separate power and data lines [5].

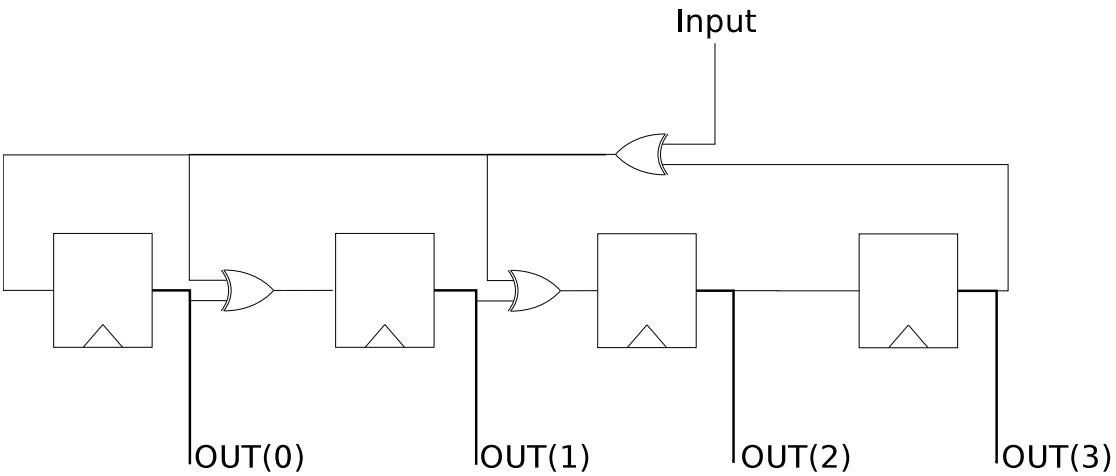

Figure 7: A four bit LFSR with binary generator polynomial of `1101`, the polynomial dictates where the XORs should be placed. The leftmost register is referred as the first register and the rightmost one is referred to as the last

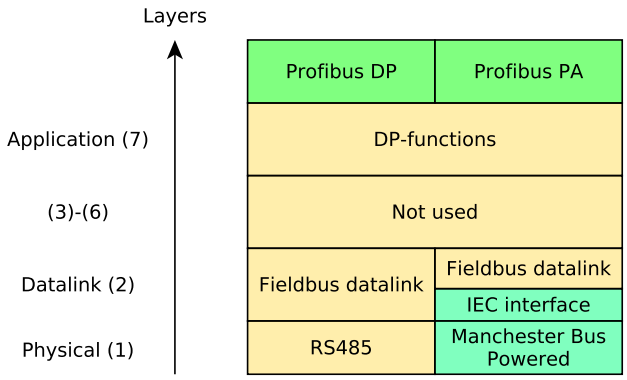

Figure 8: The OSI layers used by Profibus DP and profibus PA.

Like most fieldbuses, Profibus uses a master/slave structure where a master sends out requests on the bus to one of the slaves and the addressed slave answers. A slave cannot send a message over the bus unless a master first makes a request. A bus network consists of one or more masters and one or more slaves.

A Profibus master has several tasks, ranging from finding new devices on the bus to detecting failure of slaves. As there can be several masters present on the bus a token has to be passed between the masters to decide who shall control the bus [27].

The messages on the bus can either be sent cyclically or acyclically. When the messages are cyclic the master sends a message to each slave consecutively, even if the

master is not configured to communicate with that address. When the messages are acyclic they can be sent to any slave.

2.4.1 Profibus DP

There are three versions of Profibus DP: DPV0, DPV1 and DPV2. The main difference between the versions are what commands they support. DPV0, the first version, supports only cyclic data exchange with the slaves. DPV1 adds the support for acyclic data exchange. DPV2 is not an official version, but instead a collection name of extensions of the DPV1 that got DPV2 as a commonly used name [27].

Each message, also called frame, that is sent over Profibus DP is one of five possible messages. There are four frames used for transfers of data or diagnostics, and one frame used for acknowledgement. The frames are shown in Figure 9. In the figure, SD is the start delimiter and indicates what type of frame that follows. DA is the destination address and SA the source address. FC is the function code, depending on if the message is a request or a response, the function code will be handled differently. FCS is the Frame checksum, the checksum is calculated by adding all previous bytes except the SD. Data are bytes containing information either to write or read configuration, or to write or read from the actuator or sensor. LE and LEr are the length of of the frame, starting counting from DA and ending at Data. The length is sent two times as an error in the length would confuse the receiver of the message and cause large delays due to a device waiting for data to finish when there are none. This could be bad if the bus is used in a real-time system.

As can be seen, all frames but the short acknowledge starts with a start delimiter. There are four different start delimiters depending on what message is to be sent. The delimiter tells the receiving device how the following bytes should be interpreted. Unless the start delimiter is SD2, the start delimiter determines the length of the frame [28].

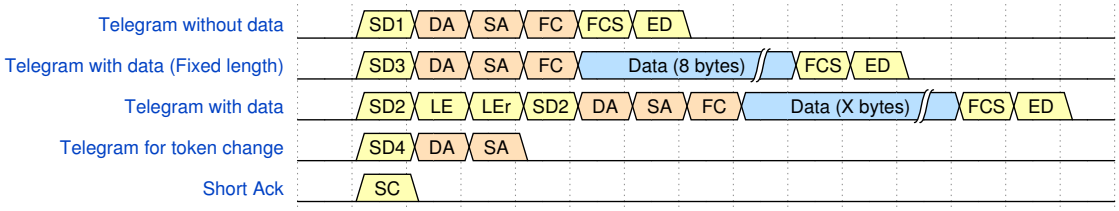

Figure 9: The different message frames allowed for Profibus DP. Yellow bytes indicate that the byte is not included in the checksum. Orange indicates the byte contains addresses and function for the receiver. Blue indicated that the byte is transferring data to or from the slave.

When sent, each byte is encoded into a Profibus character to be sent over the physical layer. The character contains 11 bits for every byte sent, using three extra bits. The extra bits are: a start bit, a parity bit and a stop bit, an example of a byte is shown in Figure 10. When there is no activity on the bus, the bus is held high. When a transfer is initiated the start bit will then come, and it being a 0 will indicate activity on the

bus. The 8 bit data is transmitted followed by a 1 bit even parity. The character ends with a stop bit with the value 1.

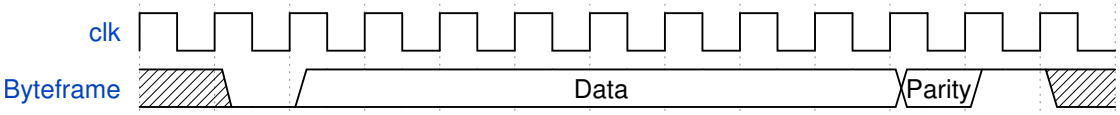

Figure 10: Structure of a Profibus DP character for sending one byte of data. The byte begins with a start bit, and ends with a parity bit and end bit

2.4.2 Profibus PA

The physical layer of Profibus PA is a DC voltage with Manchester encoded data superimposed upon it. Profibus PA uses a two-wire differential interface, meaning that the system only can send in half duplex. Since the data is sent with Manchester encoding it is possible to retrieve the clock at the receiving side and to take difference the data from the DC voltage from the signal. The data is sent over the physical medium using current, a 1 is +9 mA and a 0 is -9 mA compared to the idle current. A slave shall always drain at least 10 mA when idle. The Profibus PA frame is similar to the DP frame described in Section 2.4.1, but with some changes:

- A Profibus PA frame starts with between 2 to 8 bytes of preamble.
- After the preamble, a 1 byte Start Of Frame (SOF) is sent.
- The checksum is replaced by a 2 byte CRC.
- A frame ends with a 1 byte End Of Frame (EOF).

Profibus PA supports the different frames used in Profibus DPV1. The frames used in Profibus PA are displayed in Figure 11.

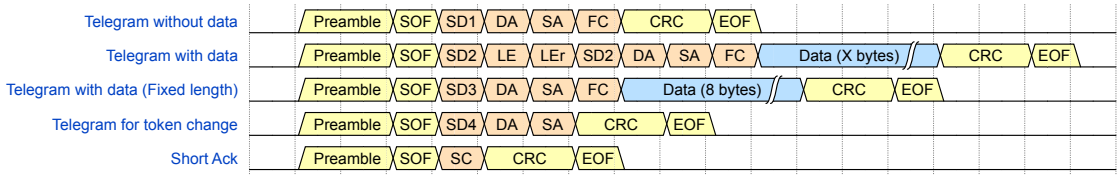

Figure 11: The different message frames allowed for Profibus PA. Yellow bytes indicate that the byte is not included in the CRC. Orange indicates the byte contains a start delimiter, address or function for the receiver. Blue indicated that the byte is transferring data to or from the I/O.

The Preamble byte is alternating 1s and 0s allowing for the receiver to synchronize the Manchester characters. The start frame and end frame use the N+ and N- characters to create a unique message that can never occur in the data. These three messages are shown in Figure 12 [5, 6, 22].

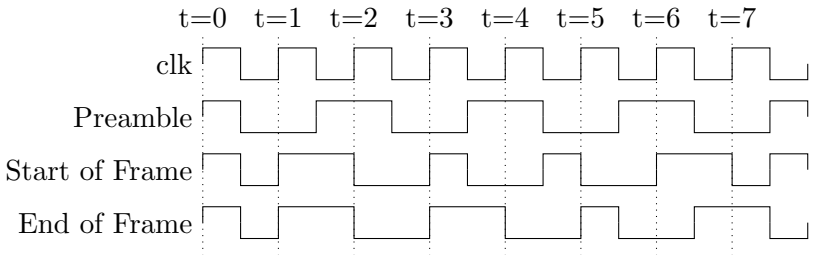

Figure 12: The three special Profibus PA bytes

The two bytes CRC in Profibus PA uses `0x1DCF` as its generator polynomial, which gives a HD = 5 for messages shorter than five octets, and HD = 4 for messages under 344 octets. The longest Profibus PA frame is 256 bytes so the HD = 4 will always be true [5].

Power over Profibus PA supports several different setups, called ”types”. What type to use depends on where the bus is used: for use in explosive environments 13.5 V is used, while for other applications 24 V or 31 V are used. The type also dictates how much power that can be supplied by the bus, the most being 31 W. [22]

2.4.3 Profibus DP vs Profibus PA

As mentioned in Sections 2.4.1 and 2.4.2, the difference between the DP and PA standards lies in the physical medium and in the frame structure. Figure 13 shows the difference in frame structure of a standard message for the respective buses. The Profibus PA requires more bytes to be sent due to the need of a preamble, start delimiter and end delimiter. In contrast, each byte sent using Profibus DP adds a start bit, parity bit and end bit [26].

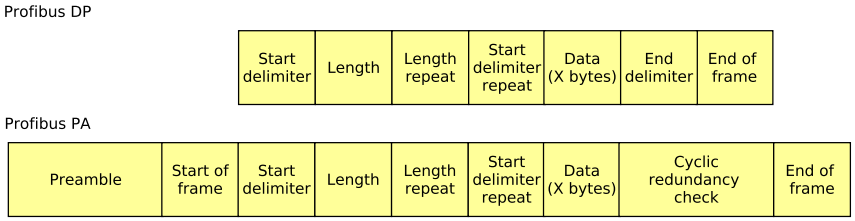

Figure 13: The frame structure of Profibus DP and Profibus PA.

The Profibus DP and Profibus PA also use different, but similar, cables: the Profibus DP cables are thinner than the Profibus PA cables. The Profibus DP cables have a diameter of ≈ 0.60 mm whereas Profibus PA cables have a diameter ≥ 0.80 mm. The thicker cables give a lower resistance, reducing the voltage loss, allowing more power to be supplied over the cable. The thicker cables also mean that transmissions over longer

distances than described in Section 1.2 is possible due to lower resistance these cables provide [29, 30].

2.5 AMIS-492x0 Chip

The AMIS-492x0 chip converts current signals of the Profibus PA into logic voltage levels used by the FPGA. The chip is mostly analog in design and requires a number of passive components to function. The AMIS-chip has two different voltage supplies: digital supply, V_{DD},and analog supply, V_{CC}. Depending on how the passive components are connected the chip can either receive power from the Profibus PA, or from an external source. During the project both of these solutions were used. As mentioned in Section 2.4.2 the slave has to drain a current $\geq$ 10 mA, and the signals being $\pm$9 mA, with the AMIS-chip this is done by controlling the draining of a current through a transistor controlled by the AMIS-chip [12].

A schematic of the receiver part of the design is shown in Figure 14, which includes both components inside the AMIS chip and external components. The schematic is based on the one found in the AMIS-492x0 datasheet [12]. First the signal is separated from the DC on the bus and instead biased around 2 V. The voltage is biased due to the internal Operation Amplifiers (OP amplifiers) having a range between 0 and 5 V. Two diodes are added to prevent too high or low voltage from damaging the IC. Secondly the signal is filtered through a bandpass filter. Using the recommended values yields a pass band between 1 kHz and 47 kHz. The signal is then compared to 2 V to determine if the signal is high or low, this operation has an hysteresis of 40 mV to prevent glitching due to noise. Finally the signal is converted to the digital voltage levels V_{DD} and outputted on the Receiver Data (RXD) pin. There is also an Receiver Activity (RXA) pin indicating there is activity on the bus.

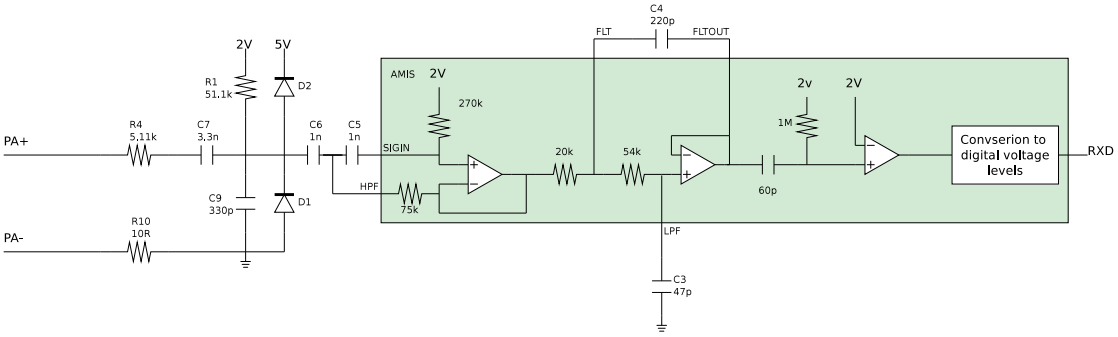

Figure 14: Schematic of the receiver part of the recommended design from the AMIS-492x0 datasheet [12]. The green area of the schematic represents components inside the AMIS-chip.

A schematic of the transmitter part of the design is shown in Figure 15. As in the receiver figure this figure includes both components inside the AMIS chip and external components. When the Request to send (RTS) is high no data is transmitted, when it

is low either a 1 or 0 is transmitted depending on the value of Transmitter Data (TXD). Depending on the input values the AMIS-chip will output the current for one of three output symbols: Idle, high and low. First the two digital inputs are handled by logic to determine what symbol to be sent. Second, an OP amplifier turns the three symbols into different voltages: 2.9 V represents a 0, 2.5 V represents idle and 2.1 V represents a 1. This stage also limits the rise and fall time of the signal using a capacitance to control the slew rate. Finally an OP amplifier controls a transistor to generate the output current.

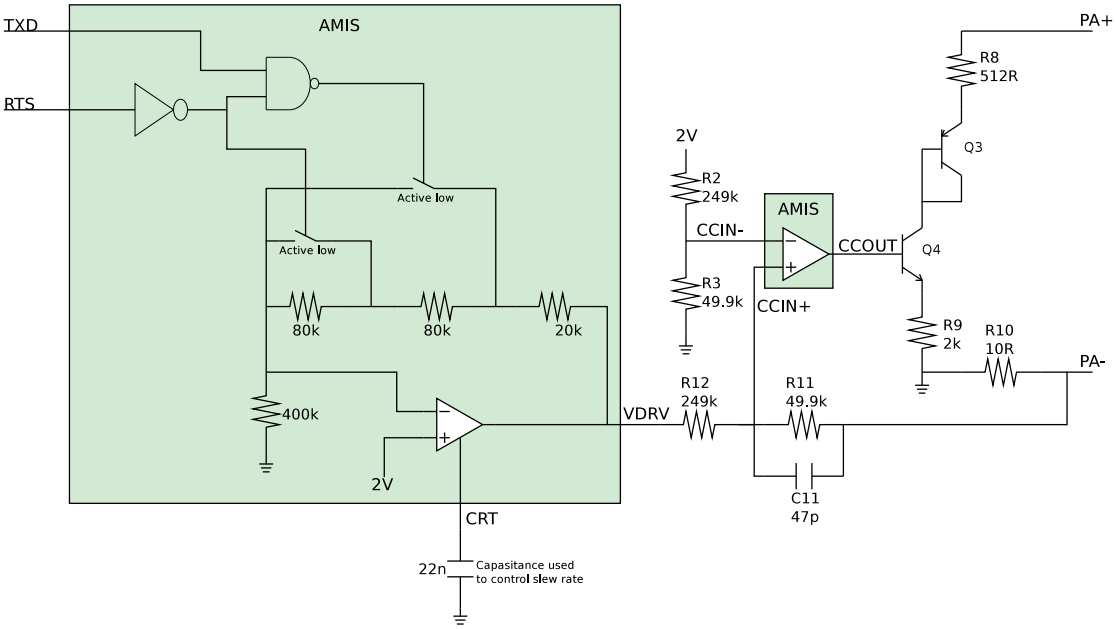

Figure 15: Schematic of the transmitter part of the recommended design from the AMIS-492x0 datasheet [12]. The green area of the schematic represents components inside the AMIS-chip.

3 Method

To complete this project we will use an implementation approach combined with a verification method, both described in this chapter. First the materials used are described in Section 3.1, then procedure of the implementation is described in Section 3.2, followed by how the verification was done in Section 3.3.

3.1 Materials

Several hardware and software tools were used to complete the project. How these tools were used is described in this Section.

3.1.1 ABCC

An ABCC module for Profibus DPV1 was modified and used to access the signals from the NP40. The components on the PCB were removed to be able to connect an AMIS-492x0 [12], the function of this chip is presented in Section 2.5. The programming of the FPGA fabric and the code running on the CPU were also changed during development.

3.1.2 Libero SoC

Libero SoC is Microsemi's suite for HDL synthesis [31], and it contains several useful tools for different parts of development for Microsemi's devices. The suite itself served as a hub for different features: an interface for including Intellectual Property (IP) cores, placing and routing, programming and timing analysis. Worth mentioning is that it does not automatically verify the timing when doing the place and route [31].

Modelsim ME was used to simulate separate HDL components as well as connected components. Both `.do`-files and testbenches were used to verify functionality. The `.do`-files were used to provide initial stimuli to see if the component worked at all, and when this was verified it was connected to other components and tested using a testbench [32].

Synopsis Synplify Pro was used to synthesize HDL code into a netlist of FPGA elements. It was also used during development to create schematics of the HDL code for the ABCC DP-V1 to understand how subcomponents were connected [33].

Synopsis Identify was used to create an on-chip-logic analyzer. This analyzer was used to monitor states and make sure the synthesized code behaved as expected [34]. debugging the Smartfusion2. The software is based on Eclipse [35].

Communication with the SmartFusion 2 SoC was done with a Flash Pro 4 Joint Test Action Group (JTAG) [36] hardware programmer. The communication mainly consisted of programming the logic, but also transferring compiled C-code to the processor and debugging the design.

3.1.3 ABCC Development tool

ABCC Development tool was used to handle the data from the Anybus. The program connected with an ABCC-40 slave using a USB interface. With this program it is possible to simulate input and outputs of the Profibus PA slave. This program also has functions such as reading and writing NP40 parameters data, e.g. slave address.

3.1.4 Profibus network

A Profibus network was set up to verify the functionality of the Profibus PA slave. This system consists of a Programmable Logic Controller (PLC) acting as a Profibus DP master and a coupler converting the Profibus DP into Profibus PA. For the DP side Profibus DP cables were used and for the PA side two wires were used. The two units were driven by 24 VDC supplied by a power supply.

Table 1: Hardware used to create the bus network

PLC	DP/PA Coupler
Siemens Simatic S7-300 CPU315-2 DP	Siemens Simatic 157-0AC81-0XA0
Siemens Simatic S7-1200 CPU1212C	

The PLCs act as a Profibus DP master and was configured using Siemens Step 7 [37]. The configuration was set up to read cyclic messages from PA slave using address 11. The Simatic S7-300 [38] was used in early development, the PLC was not programmed with any code, resulting in it polling all addresses to look for slaves, and setting configuration on the slave on address 11. When the full system verification was done a Simatic S7-1200 [39] was used. This PLC was programmed to return the slave input data as output to the slave.

The DP/PA coupler works as a transparent converter between the buses, which means that the bus master cannot detect or interact with the coupler in any way. The coupler only repeats the messages on the first bus on to the other bus, e.g., from Profibus DP to Profibus PA . The coupler also filters out traffic to keep the bus as silent as possible. One example of messages filtered is if a DP master contacts a DP slave: the request will be passed through to the PA bus as the coupler does not know if the slave requested is on the PA or DP bus but the response from the DP slave is not passed through the coupler as this information is of no use on the PA side [40].

3.1.5 Profitrace

Profitrace [41] is a software tool for monitoring bus activity on a Profibus DP or PA bus using the Proficore hardware. Profitrace was used to verify bus activity, as well as detecting what errors that occurred on the bus. Depending on what probe that was used either DP or PA can be monitored. For Profibus DP the bus speed can either be detected or set manually.

The information Profitrace delivers includes source, destination, and other useful information of the frames sent on the bus. The program can record a large amount of messages to verify that functionality holds high reliability [41].

3.1.6 Logic analyzer

Logic analyzers were used to analyze the output from the AMIS chip. The output was also recorded and converted into HDL to provide input for the testbench. Two different logic analyzers were used, one for recording long strings of data with low sample rate, and one for logging short strings of data with higher sample rate, giving a more accurate representation of the signal.

3.2 Procedure

The thesis started with a brief literature study of the components available, i.e. the AMIS chip and the NP40 processor, as well as of the standards used.

With knowledge of the components a breakout board for the AMIS chip was created and connected to an ABCC-40. The PCB schematic followed the suggested schematic from the datasheet. With the PCB assembled the output of the AMIS-chip was verified using a logic analyzer with built-in Manchester decoding. These values were compared to the expected values read from the Profibus analyzer.

The next step of the development was to decode the Manchester encoded Profibus PA signal in the NP40. The HDL was tested in Modelsim using data sampled using a logic analyzer. When the HDL worked in simulations it was synthesized and downloaded into the NP40. The NP40 was connected to the AMIS chip to read messages from the Profibus PA via the AHB sent from a Profibus DP-V1 master, through a DP/PA coupler to the AMIS chip. To help with debugging at this stage a logic analyzer and a Profibus PA analyzer were used.

With data successfully received, an HDL module was written to interface the decoded bus data with the existing Profibus DPV1 core. This was done through a bridge with a state machine controlling the inputs to a First Input First Output (FIFO) buffer converting the PA frame to a DP frame. The Profibus DPV1 core was then debugged using an on-chip logic analyzer. When the core handled the data correctly, the transmitting HDL was designed. When the transmitting HDL was complete the output was verified first by using an oscilloscope and then by using the bus analyzer.

As mentioned in Section 1.3, if there was time a PCB would be designed in accordance to the ABCC form factor. This work was done using Cadence OrCAD and used the Profibus DP-V1 ABCC schematics and board design as the starting point, to reduce the work necessary.

3.3 Verification

To verify the function of the system and the subsystems tests and verification were performed. There were two kinds of verification, in software and in hardware.

3.3.1 HDL Simulation

Each HDL component was tested individually to make sure it worked as intended before it was connected to a larger block or system. For larger modules testbenches were developed to verify that the modules worked as intended. These testbenches were fed sampled values and the response was checked and verified.

3.3.2 Hardware Verification

When the modules worked as intended in simulation they were synthesized and tested with the hardware. Using the built in debug features of the NP40, values were monitored: either by buffering frames and then reading the stored frames from the AHB bus by using an on-chip-debugging, or by writing to AHB registers to control certain parameters. Tools such as a Profibus PA analyzer were also used to verify bus inputs and outputs.

3.3.3 Full System Validation

The final system was verified by using a PLC acting as a DP bus master. The PLC reads values from the slave and then sends the read value back to the slave. The DP bus master will then send a frame to a DP/PA coupler and the frame was converted back into a Profibus PA message. The PA message was separated from the DC voltage and the signal voltage levels converted to CMOS levels by the AMIS chip. The NP40 then decodes the frame and translates the message to the Anybus standard. The Anybus interface signals were then translated to USB and sent to the ABCC Development tool running on a Personal Computer (PC). Using a function in the ABCC Development tool to validate the design by simulating input values and then comparing these to the values the master sets at output, i.e. the master repeating the data. The ABCC Development tool then compared the data to verify that it was sent without any errors.

A Profibus analyzer was also used to verify that the slave did not send when it should not or if the slave was sending incorrect frames. The analyzer was used to gather data while the master and slave were communicating. It was also used to verify that the slave did not answer to other addresses. The setup used for the full system verification is shown in Figure 16. The current Profibus DP-V1 slave was verified in the same way, save the DP/PA converter.

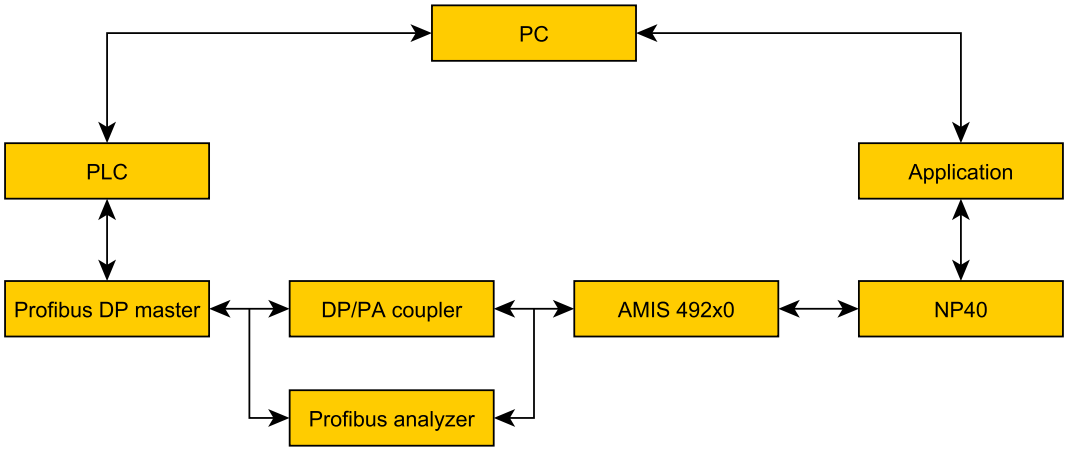

Figure 16: A schematic over how the full system test was performed

4 Implementation

This chapter describes the implementation process of the thesis. It starts with how the HDL was designed and functions in Section 4.1, followed by how the PCB was designed in section 4.2.

4.1 HDL

The HDL was built around the Profibus DPV1 implementation. To reduce the time and effort needed, as much of the original DPV1 implementation as possible was left unchanged. The developed implementation therefore converted the PA frame to a DP frame, meaning that the only change between the DP and PA code would be the physical interface. This way of implementing was not the most effective solution in terms of speed, power or area, but it saves on development and verification time. The HDL modules implemented are displayed in Figure 17.

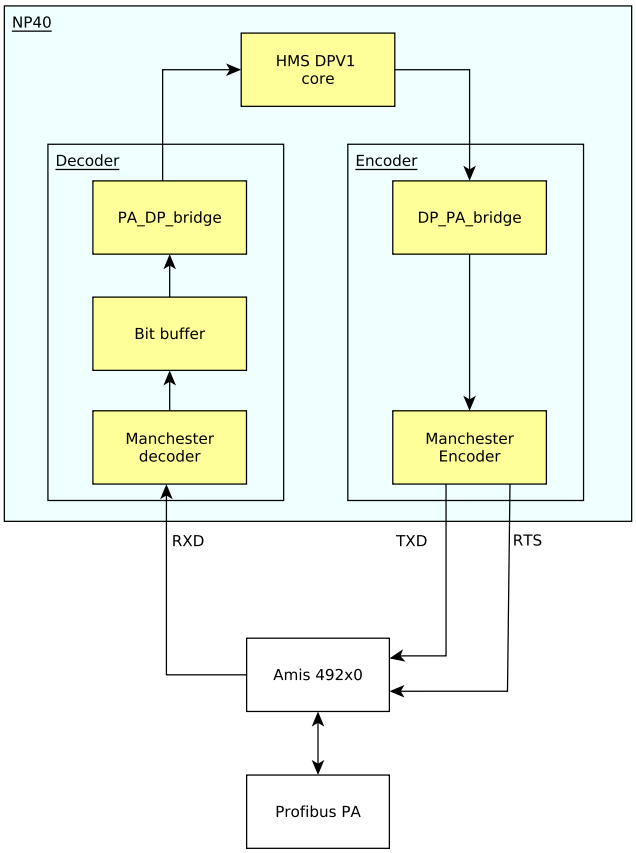

Figure 17: Overview of the modules used to interface the AMIS with the current HMS Profibus core

4.1.1 Decoder

The decoder receives a signal from the external IC to decode and converts it into bytes to be read by the Profibus core from the Profibus DPV1 implementation. When the decoder was designed, different layers were considered and the sub-blocks do different stages of the decoding. The Manchester encoded signal from the AMIS-chip is first decoded into bits, these bits are then buffered until a whole byte is read. The bytes are then used to calculate the CRC, translated into a DP frame and stored in a FIFO buffer. The difference in frames is described more thoroughly in Section 2.4.3. The decoder consists of three blocks: `the Manchester decoder`, `the bit buffer` and a `PA_DA_bridge`. Figure 18 shows how these components are connected, and what signals are used to interface them.

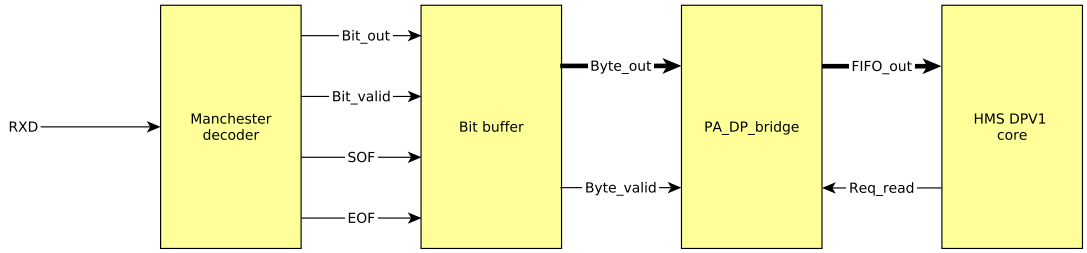

Figure 18: Overview of the blocks used in the decoder and how these interact with each other.

The decoder is controlled by an enable signal from the DPV1 core. This signal is low until an edge is detected in the RXD. When the edge is detected the enable signal goes high either until a full frame has been received or the core decides that a timeout has occurred.

4.1.1.1 Manchester decoder

The `Manchester decoder`'s task is to decode an incoming Manchester encoded signal into raw bits. The AMIS chip supplies two signals, the enable signal RXA and the data signal RXD. The RXA is not used, as an edge in RXD indicates activity on the bus. The RXD is sampled two times for each bit to detect if the transition is from 0 to 1 or from 1 to 0. The sampling is shown in Figure 19. The decoder also resynchronizes the sample clock on each edge, making it robust against frequency errors. As Manchester code only holds one data-line the incoming data have to be synchronized to make sure the signal is sampled correctly. This synchronization is done at the preamble stage using the Manchester code property that there has to be an edge in the middle of each bit. If a bit is detected and the two samples are the same, the decoder delays the sampling half a bit and then checks if the the new sample is different. When 8 correct Manchester symbols have been found the decoder locks the synchronization to prevent the SOF byte to desynchronize as it contains Manchester characters without an edge.

An example of how the sampling may look like when synchronizing can be described by Figure 19. If the first two samples are s1 and s2, they will represent an incorrect Manchester symbol as these samples have the same value. The `Manchester decoder` then keeps one of the samples and wait for next sample, s3, which is a 0. As the two stored samples now are different there is a correct Manchester symbol present. The decoder then samples the next two bits, s4 and s5, and these are also the different, so this synchronization seems to be the correct.

A separate solution is also implemented for detecting the SOF and EOF. This solution stores all sampled bits in a shift register and then compares the contents of the shift register with the expected SOF and EOF. The `Manchester decoder` outputs a sampled bit, and a signal indicating when a bit is valid. The detection of the SOF and EOF is also outputted.

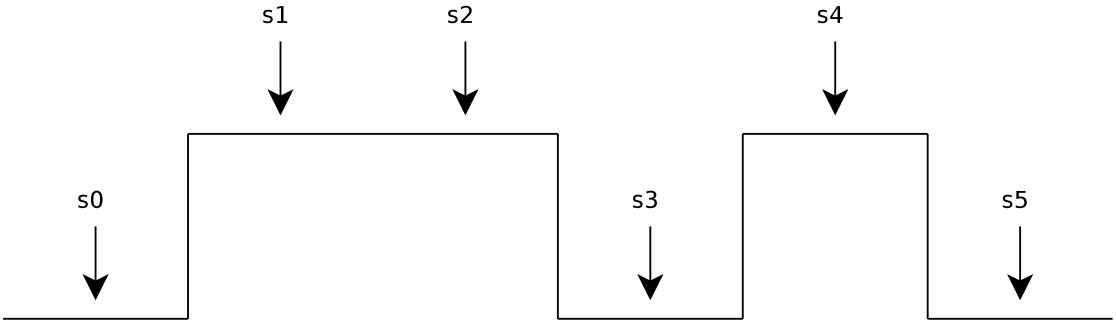

Figure 19: The sampling of the incoming Manchester encoded signal. Each bit is sampled two times to determine the edge.

4.1.1.2 Bit buffer

The `bit_buffer` gathers the incoming bits into bytes, Figure 20 shows how this module functions. When idle the `bit_buffer` waits for an SOF to be detected. When this happens the next bit will be the first bit in the "real" message. The buffer then records bytes until either the enable goes low, or the EOF is detected. When either of these changes happens the `bit_buffer` goes back to waiting for a new SOF. The `bit_buffer` outputs the output byte and a signal indicating that the value of the byte is valid.

4.1.1.3 PA_DP_bridge

The `PA_DP_bridge` has two tasks: to translate the PA frames into DP frames, and to store these translated frames until the DPV1 core reads them. The storage part of the PA_DA_bridge is built upon a FIFO buffer used in the DPV1 HDL to ensure the DPV1 core works as usual. The buffer has a depth of four bytes, but is rarely getting full due to the DPV1 core operating at a much higher frequency than the Profibus PA input, if the FIFO gets full, a byte of the frame will be lost and the frame will be corrupt. The input of the FIFO is controlled by a state machine, whose function is shown in Figure

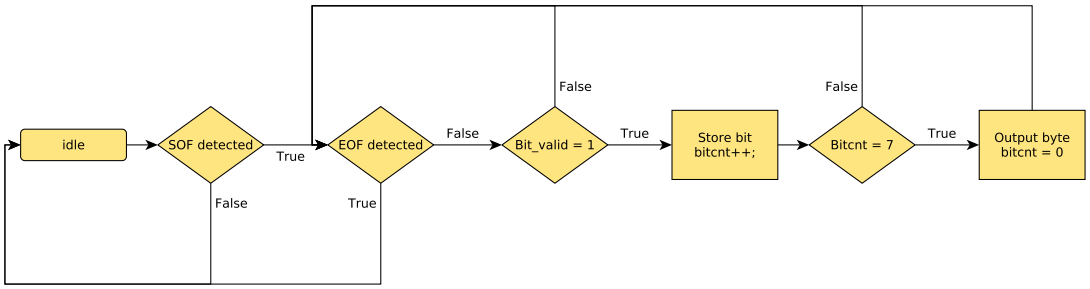

Figure 20: Functional design of the `bit buffer`. All blocks are evaluated at each clock-cycle

21 detecting the start delimiter of the frame, and from that the length of it. The length is needed to know how many bytes to be included in the CRC calculation and when to insert the FCS and ED needed for the Profibus DP frame. When the enable signal is low the FIFO empties the buffer and goes back to the initial state to wait for the first byte in the next frame.

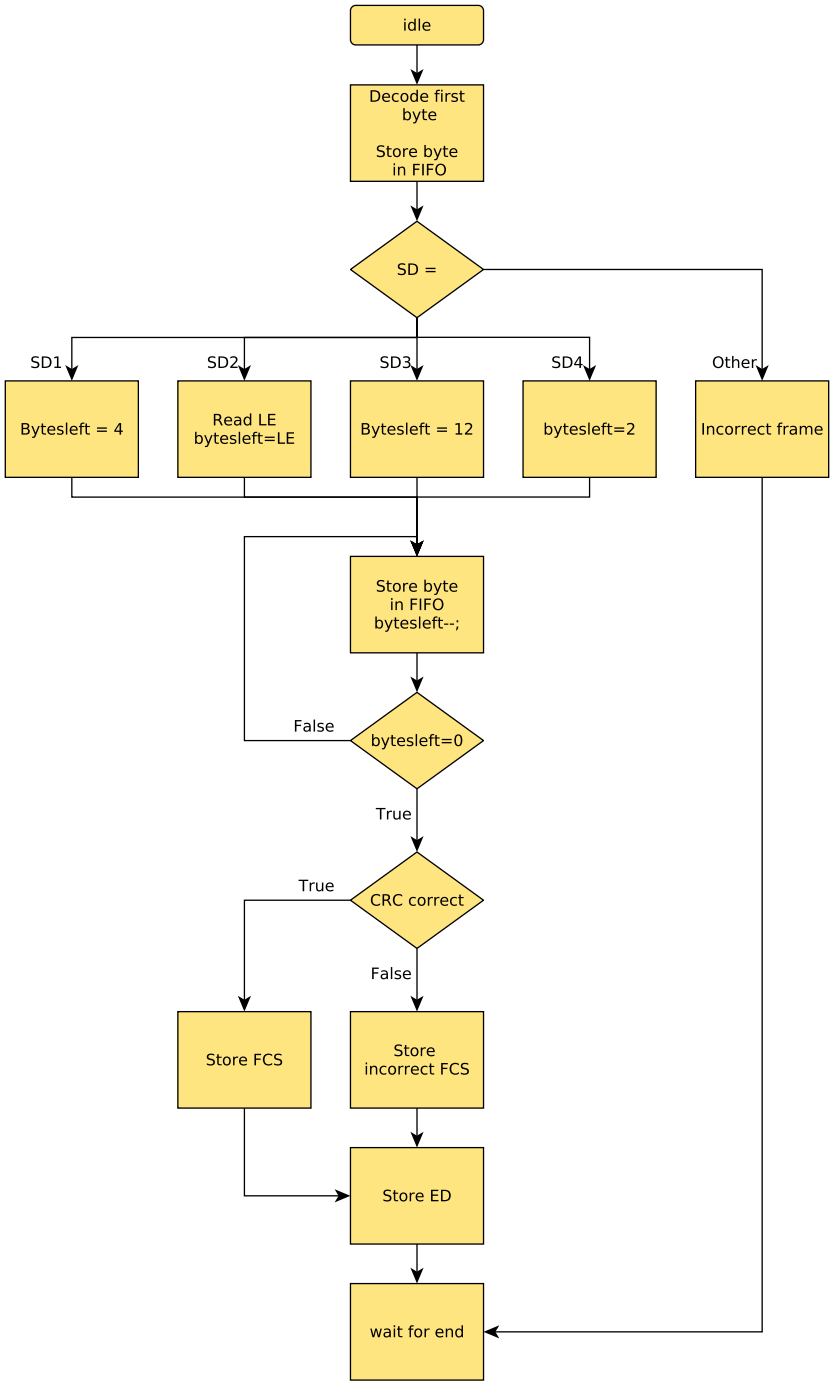

Figure 21: Functional design for the PA_DP_bridge.

4.1.2 Encoder

The encoder uses the values sent out from the Profibus core and converts them to Manchester-encoded data. The encoder outputs these data to the AMIS-chip along with a signal indicating that the chip should transmit data over the bus. Like the decoder, the encoder was designed in layers but as the function is simpler: only two modules were needed. The encoder consists of two blocks: `DP_PA_bridge` and the `Manchester_encoder`. Figure 22 shows how these blocks are interfaced towards each other.

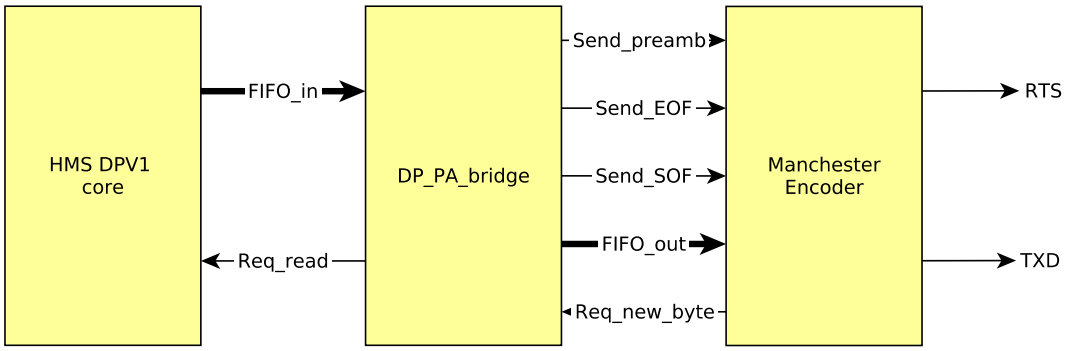

Figure 22: Overview of the blocks used in the encoder and how these interact with each other.

4.1.2.1 DP_PA_bridge

The `DP_PA_bridge` works much like the `PA_DP_bridge` in the decoder. It contains a FIFO to receive values from the DPV1 core. The contents of the buffer is then read into a state machine for the frame to be translated from DP to PA. This state machine is shown in Figure 21. Before the first byte of the frame is sent the preamble and SOF has to be sent. In these cases, the `DP_PA_bridge` sends out signals indicating what specific byte that should be sent. The start delimiter is then read to determine the length of the frame, unless the start delimiter is SD2, then the length of the frame is determined by the LE and LEr bytes. If none of the start delimiters are detected, i.e., a short ack is to be sent, the bridge sends the CRC instead directly after the first byte. If the frame is not a short ack the data are transmitted, as well as the CRC is calculated, until the CRC is to be sent. The state machine ignores the checksum sent from the master, as no error should occur within the chip. The CRC is then sent, and finally the EOF is sent, like for the preamble and SOF, the EOF the `DP_PA_bridge` sends a signal indicating the EOF should be sent.

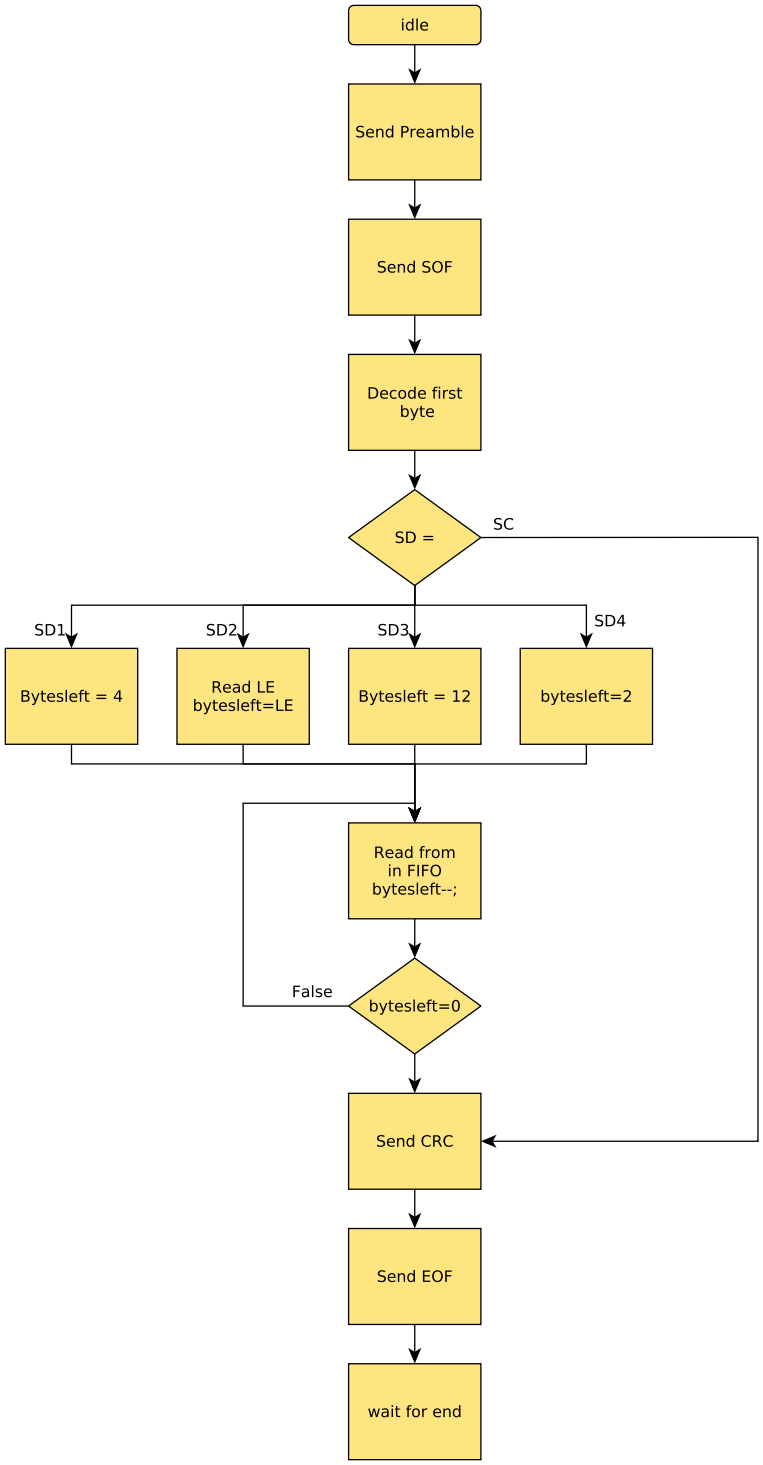

Figure 23: Functional design for the `DP_PA_bridge`.

4.1.2.2 Manchester encoder

The `Manchester encoder` receives a byte from the `DP_PA_Bridge` and encodes it into a Manchester encoded vector. This vector is then sent to the AMIS-chip. If either of the special bytes are to be sent, the output data are set to a predefined vector. The predefined vector then gets priority over any eventual data.

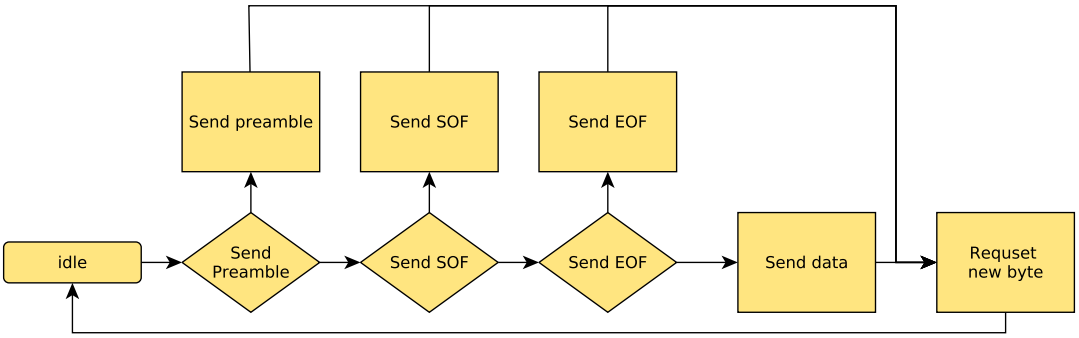

Figure 24: Functional design for the `Manchester encoder`. All blocks are evaluated at each clock-cycle.

4.2 PCB

During development three PCBs were designed. The first two were simple boards for connecting the bus, the AMIS-chip, and the NP40 together, and the third PCB was a final ABCC-PCB in the ABCC form factor.

The first PCB was designed according to the example design from the AMIS-492x0 datasheet [12]. As this was a prototype the size of this PCB did not matter, but the size was chosen to be large to allow for the possibility of later modifications. Once assembled this design was used for most of the development of the HDL. This PCB was directly connected to the NP40, as opposed to using optocouplers for galvanic separation.

The second PCB was designed after a working prototype had been created. The main goal of this version was to evaluate how the AMIS-chip behaved with power supplied from the ABCC instead of the from the Profibus PA. Supplying power from the ABCC, several components could be removed from the design allowing it to fit the small ABCC form factor. Unlike the first board, both the analog and digital power were supplied with 5 V from the ABCC. Figure 25 shows the revised circuitry between the AMIS-chip and the Profibus PA for the second PCB. The output of the AMIS-chip was connected to optocouplers to separate the Profibus PA and the Anybus providing galvanic isolation. In this version R9 from Figure 15 was changed to a diode D3 and a 100 Ω resistor R13 as the template design could not drain as much current from the bus with the 2 kΩ resistor. Another change compared to the schematic shown in Amis-492x0 datasheet [12] is that the circuits for providing 5 V supply are removed.

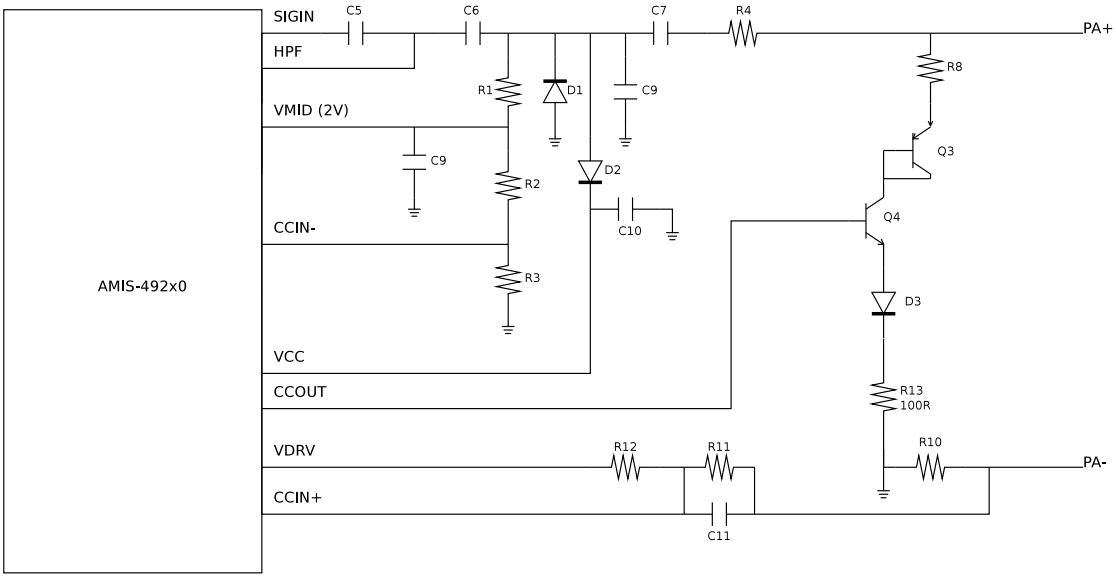

Figure 25: Schematic of the discrete components of the second version of the PCB.

Finally an ABCC PCB was designed. Compared to the previous two PCBs, this design was constrained to the size of the ABCC form factor. The ABCC-PCB uses the same components for the bus-side as the second prototype, and the layout on the host

side is the same as in the Profibus DPV1 design. The ABCC uses a 4 layer PCB: the top and bottom layer are used to connect the components and the two middle layers serve as ground and power planes respectively. This PCB was never manufactured during the project, so the design serves as a proof of concept for the possibility to fit the form factor.

5 Results

This chapter shows the outcome of the thesis. A Profibus PA implementation for the ABCC-40 module was successfully completed. First the results from the HDL synthesis are presented in Section 5.1, followed by the PCB in Section 5.2, and finally the result of the verification and validation is presented in Section 5.3.

5.1 HDL synthesis results

Table 2 shows how many FPGA primitives the Profibus PA and DPV1 implementation uses. Here only the physical interface is synthesized as this is the only difference between the two versions. In Table 2, Logic Element is the total number of logic elements used, 4LUT is the use of a 4 input LUTs, and DFF is the use of a flip flops.

Table 2: Number of used FPGA resources for the new Profibus PA implementation and the old Profibus DPV1 implementation

Type	PA	DP	Change [%]
Logic Element	765	286	167
4LUT	722	239	202
DFF	289	179	61

As can be seen the new implementation is much larger than the original one. One reason for this is that the new Profibus PA implementation has to translate the frame to Profibus DP before sending it to the DPV1 core.

When running the place and route at the target frequency of 48 MHz, the frequency used in the DPV1 core, the Profibus PA physical layer the tool reported that the interface could be run at 148.06 MHz. The same number for the whole Profibus PA implementation is 99.1 MHz, so the new physical interface does not limit the maximal frequency.

Using the power analysis tool in Libero the power consumption of the SoC can be estimated. For the whole system this number was 333 mW. This number does not include the external components of the PCB such as the 5 V supply on the bus side.

5.2 PCB

A final implementation of the Profibus PA module was possible to implement in the ABCC-40 standard. It is possible to fit the final circuit design into an ABCC form factor board. The component placement of the ABCC PCB is shown in Figure 26. The left layout shows the top side of the PCB, to the left the NP40 can be seen, to the right of the NP40 are the optocouplers and then on the right edge is a connector for Profibus PA. The right PCB shows the bottom side: on the left edge is the Anybus contact, in the middle the AMIS-chip can be seen and around the backside of the connector are the discrete components shown in Figure 25. The components on the PCB consumes 78%

of the available area. The PCB was not completely able to fulfill the ABCC standard as some components are placed too close together.

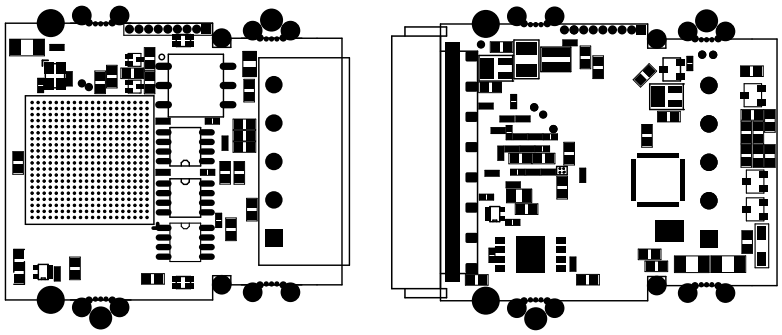

Figure 26: The final Profibus PA ABCC designed in the project. The right sides of the PCBs are the isolated fieldbus side.

5.3 Verification

This section describes how the implementation was verified, starting with software verification using Modelsim and then the results of the full system verification.

5.3.1 HDL Simulation

All of the HDL were simulated as a module with sampled data from the bus to verify the function of the decoder. The encoder was then connected to the decoder to be able to first encode and the decode frames. Both these tests were successful, proving that the decoder and encoder worked respectively.

The `Manchester decoder` was thoroughly verified using a testbench. During this verification, the decoder could decode Manchester encoded signals with a large frequency error. In software testing the frequency could deviate down to 33% lower and 23% higher than 31.25 kHz without any errors in decoding.

5.3.2 Hardware Verification

With functioning HDL in simulations the design was connected to the AHB. The data read through the bus were then compared to expected frames to verify that the decoding was working as intended. During these tests the decoder was monitored by the on-chip-debugging to verify that it functioned as it should.

5.3.3 Full system validation

The full system validation was done as described in Section 3.3.3. During the validation the number of input and output bytes of the slave were set to the maximal length of 244 [6]. Each of these bytes was then read by the master and sent back to the slave, meaning

that one failing bit would generate an frame error. The looped back data were compared to the sent date by the ABCCdevtool software running at a computer. The validation lasted for 45 hours and during this time data were read and written from the slave ≈ 3.6 million times, transmitting ≈ 916 MB of frames, of which ≈ 880 MB were data. During this validation no transmission errors were detected by the ABCC development tool.

Profitrace detected 13 frame errors during the test. Twelve of these were short frames with a length of 1 or 2 bytes: the cause of these errors is probably due to noise on the bus. These errors should not cause any faults, as they were not following the frame structure. The last frame error is part of a whole frame and the cause for this is unknown. As the frame error did not cause any error detected by the ABCC development tool the conclusion is that the DP/PA coupler and Profibus PA concept module have better algorithms to decode the PA signal than the Proficore. This theory is strengthened by observations during prototyping, when the DP/PA coupler could turn PA frames to DP, but the Proficore reported illegal frames.

6 Discussion

The ABCC-40 module for Profibus PA has been verified in an ad hoc fashion using simulated traffic between a Profibus DP master and a PA slave, using a DP/PA coupler to translate traffic. During these tests the focus was on testing the physical layer, as the other parts of the design were unchanged and already verified. The PCB has to be updated to fulfill the ABCC form factor before the module can be considered a finalized product. The changes to be made are changes regarding spacing between components. Some future work with the module developed may be to either convert it into a bus powered version or to create a FOUNDATION Fieldbus version, using the same PCB, but rewriting large parts of the HDL and software.

6.1 Bus powering

As continued work a version of the PCB with the power supplied from the bus may be developed. Using the 31 V version of the Profibus PA it is possible to power several ABCC-40 slaves. To be able to replace already active slaves this may be a deal breaker. If the module is not bus powered additional cabling have to be used.

The ABCC-40-DPV1 uses 587 mW when transmitting at 9600 baud. As the ABCC-40-PA module developed uses very similar hardware, this power number is a good approximation for power consumption. If the ABCC is bus powered, it also has to power the host CPU controlling it. There will also be a power loss in the DC/DC converters, as these do not have a 100% efficiency, so in total the unit may use as much as 1 W. Assuming this power consumption it is then possible to use up to 31 devices on a Profibus PA segment.

Since a bus powered ABCC-PCB would not have the need for any galvanic isolation, this frees some area to use for the DC/DC conversion.

6.2 FOUNDATION Fieldbus

Profibus PA shares the physical layer with FOUNDATION Fieldbus, i.e. Manchester encoded bus powered signals transmitting at 31.25 kbaud, meaning the physical layer can be reused for a future implementation. The FOUNDATION Fieldbus is maintained by Fieldbus Foundation and is specified in the IEC 61158 [6]. The FOUNDATION Fieldbus uses a peer-to-peer protocol, unlike the master-slave used by Profibus PA, which enables any unit in the bus to initiate messages. This can be useful when alarms are sent, instead of waiting for the master to check the slave the alarm will be sent instantly. This alarm can then be detected by all units on the bus, whereas in a Profibus PA system the master will initiate all communication with slaves [6, 42].

7 Conclusion

An ABCC-40 module for Profibus PA has been developed and is operating as intended. The module can be connected to any already present Anybus interface. The Profibus PA is functional with short cables, however, tests with longer cables have yet to be done. Since the implementation was small enough to fit on the NP40, most of the HDL modules developed can be implemented on any FPGA platform. As Profibus PA and Profibus DP only differ in the physical layer, the HDL code of an ABCC-40 module for Profibus DP was reused and, consequently, the higher OSI layers have already been verified.

When the final ABCC PCB was designed the main priority was to fit the components. As mentioned in chapter 5, almost all of the available area is occupied by components. The ABCC design could not completely satisfy the ABCC standard: the distance between the shielding and the data was disregarded due to the area limitation. If one wants to use this module this constraint has to be considered. In the current design the RXA signal from the AMIS-chip is unused, this signal has higher resistance to noise than the RXD so with longer cables this signal may be needed to ensure proper function.

References

[1] Introducing Anybus® CompactCom™ - 40 series . HMS. Accessed: 2015-01-30. [Online]. Available: http://anybus.com/products/abcc40.shtml

[2] "Fieldbus," in *Process Automation Handbook*. Springer London, 2007, pp. 365–375. [Online]. Available: http://dx.doi.org/10.1007/978-1-84628-282-9_50

[3] *HART Protocol Specifications*, Std. HCF_KIT-1, 2013.

[4] J. Powell, *Catching the Process Fieldbus : An Introduction to PROFIBUSfor Process Automation*. New York: Momentum Press, 2012. [Online]. Available: www.summon.com

[5] *Profibus Specification*, Std. European Standard EN 50 170, 1998.

[6] *Industrial communication networks - Fieldbus specifications*, Std. European Standard EN EN 61 158-2, 2010.

[7] "Electrical characteristics for balanced double-current interchange circuits operating at data signalling rates up to 10 Mbit/s," https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-V.11-199610-I!!PDF-E&type=items, 1998, iTU-T Recommendation V.11.

[8] "CAT 6 standard," http://www.cat6.com/overview/standards.aspx, accessed: 2014-11-23.

[9] "PROFIBUS – the world's leading fieldbus: easy, flexible, consistent," http://www.profibus.com/nc/download/brochures-white-paper/downloads/profibus-for-process-automation/display/, 2012, accessed: 2015-04-21.

[10] "PROFIBUS PA Systembeskrivning," http://www.profibus.se/PDF/PA_System_Description_swe.pdf, 2008, accessed: 2014-11-23.

[11] *Anybus CompactCom™ M40 for PROFIBUS*, HMS Networks, 2014, original document from HMS Networks.

[12] *AMIS-492x0 Fieldbus MAU*, http://www.onsemi.com/pub/Collateral/AMIS-492X0-D.PDF, ON semiconductior, 2013, original document from ON semoconductior.

[13] "Hardware Design Guide Anybus® CompactCom M40," http://www.anybus.com/upload/Anybus-CompactCom%20M40%20Module-7356-Anybus%20CompactCom%20M40%20Hardware%20Design%20Guide.pdf, rev. 1.34.

[14] Introducing Anybus® CompactCom™ - 30-series . HMS. Accessed: 2015-06-02. [Online]. Available: http://www.anybus.com/products/abcc30.shtml

[15] Anybus® NP40^{TM} network processor . HMS Networks. Accessed: 2015-05-07. [Online]. Available: http://www.anybus.com/technologies/network_processors.shtml

[16] SmartFusion2 SoC FPGAs Security - Reliability - Low Power. Microsemi. Accessed: 2015-01-30. [Online]. Available: http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2

[17] *AMBA 3 AHB-Lite Protocol Specification*, Std. ARM IHI 0033A, 2008.

[18] *AMBA 3 APB Protocol Specification*, Std. ARM IHI 0024B, 2008.

[19] New technology from HMS enables network connectivity for high-performance applications. HMS Networks. Accessed: 2015-05-07. [Online]. Available: http://www.anybus.com/readnews.asp?NID=162

[20] SmartFusion2 MSS Fabric Interface Controller Configuration. Microsemi. Accessed: 2015-06-02. [Online]. Available: http://coredocs.s3.amazonaws.com/Actel/SmartFusion2MSS/MSS_FIC32/sf2_mss_fic32_config_ug_1.pdf

[21] F. Williams, T. Kilburn, and G. Thomas, "Universal high-speed digital computers: a magnetic store," *Proceedings of the IEE - Part II: Power Engineering*, vol. 99, no. 68, pp. 94–, April 1952.

[22] C. Diedrich, *Profibus PA: Instrumentation Technology for the Process Industry.* Oldenbourg Industrieverlag, 2007.

[23] R. Forster, "Manchester encoding: opposing definitions resolved," *Engineering Science and Education Journal*, vol. 9, no. 6, p. 278, 2000. [Online]. Available: www.summon.com

[24] T. C. Maxino, "The Effectiveness of Checksums for Embedded Networks," Ph.D. dissertation, Carnegie Mellon University Pittsburgh, Pennsylvania, USA, 2006.

[25] A. Canteaut, *Encyclopedia of Cryptography and Security*, 2011, pp. 726–729.

[26] R. W. Mitchell, *Profibus: A Pocket guide.* ISA, 2004.

[27] M. Popp, *The new rapid way to PROFIBUS DP.* PROFIBUS Nutzerorganisation, 2003.

[28] J. Weigmann and G. Kilian, *Decentralization with PROFIBUS DP/DPV1.* Publicis Corporate Publishing, 2003.

[29] Frequently Asked PROFIBUS Questions. Procentec. Accessed: 2015-05-19. [Online]. Available: http://procentec.com/faq/profibus/index.php

[30] *Data Sheet 10/63-6.47-EN Rev. D PROFIBUS cable*, https://library.e.abb.com/public/da54d5edee606e3ac1257aa0002f3dc9/10_63_647_EN_D.1.pdf, ABB, 2012, original document from ABB.

[31] Libero SoC. Microsemi. Accessed: 2015-04-21. [Online]. Available: http://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc

[32] Modelsim ME. Mentor Graphics. Accessed: 2015-04-21. [Online]. Available: http://www.microsemi.com/products/fpga-soc/design-resources/design-software/modelsim

[33] Synplify Pro ME . Synopsys. Accessed: 2015-04-21. [Online]. Available: http://www.microsemi.com/products/fpga-soc/design-resources/design-software/synplify-pro-me

[34] Identify RTL debugger. Synopsys. Accessed: 2015-04-21. [Online]. Available: http://www.synopsys.com/Tools/Implementation/FPGAImplementation/FPGASynthesis/Pages/Identify.aspx

[35] SoftConsole. Microsemi. Accessed: 2015-04-21. [Online]. Available: http://www.microsemi.com/products/fpga-soc/design-resources/design-software/softconsole

[36] FlashPro. Microsemi. Accessed: 2015-04-21. [Online]. Available: http://www.microsemi.com/products/fpga-soc/design-resources/programming/flashpro

[37] PLC programming software for the entire SIMATIC controller range . Siemens. Accessed: 2015-06-02. [Online]. Available: http://w3.siemens.com/mcms/simatic-controller-software/en/pages/default.aspx

[38] *Datasheet: 6ES7315-2AH14-0AB0*, https://mall.industry.siemens.com/tedservices/DatasheetService/DatasheetService?control=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%3Cpdf_generator_control%3E%3Cmode%3EPDF%3C%2Fmode%3E%3Cpdmsystem%3EPMD%3C%2Fpdmsystem%3E%3Ctemplate_selection+mlfb%3D%226ES7315-2AH14-0AB0%22+system%3D%22PRODIS%22%2F%3E%3Clanguage%3Een%3C%2Flanguage%3E%3Ccaller%3EMall%3C%2Fcaller%3E%3C%2Fpdf_generator_control%3E, Siemens, 2015, original document from Siemens.

[39] *Datasheet: 6ES7212-1BE40-0XB0*, https://mall.industry.siemens.com/tedservices/DatasheetService/DatasheetService?control=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%3Cpdf_generator_control%3E%3Cmode%3EPDF%3C%2Fmode%3E%3Cpdmsystem%3EPMD%3C%2Fpdmsystem%3E%3Ctemplate_selection+mlfb%3D%226ES7212-1BE40-0XB0%22+system%3D%22PRODIS%22%2F%3E%3Clanguage%3Een%3C%2Flanguage%3E%3Ccaller%3EMall%3C%2Fcaller%3E%3C%2Fpdf_generator_control%3E, Siemens, 2015, original document from Siemens.

[40] *Bus links DP/PA coupler, active field distributors, DP/PA Link and Y Link*, https://cache.industry.siemens.com/dl/files/696/1142696/att_31151/v1/dppa_coupler_afdis_dppa_link_y_link_manual_en-US_en-US.pdf, Siemens, 2011, original document from Siemens.

[41] ProfiTrace - Mobile PROFIBUS Combi-Analyzer. Procentec. Accessed: 2015-04-21. [Online]. Available: http://www.procentec.com/profitrace2/

[42] "FOUNDATION fieldbus H1 or Profibus PA? ," http://www2.emersonprocess.com/siteadmincenter/PM%20Central%20Web%20Documents/Eng%20Sch%20-%20Buses%20301.pdf, accessed: 2015-05-21.