

Domain Engineering Sciences
Rte du Rawyl 47
CH-1950 Sion 2
Phone +41 27 606 85 11
Fax +41 27 606 85 75
info@hevs.ch

www.hevs.ch

Degree Course Systems Engineering

Option Infotronics

Diploma 2009

Carlo Arnold

Hes-so VALAIS WALLIS

Rte du Rawyl 47 - 1950 Sion 2

*Ethernet
traffic measurement point*

Professor

François Corthay

Expert

Patrik Arlos

HES-HEVS (Sion)



EM000006112734

Sion, 04.09.2009

IT / 2009 / 12

<input checked="" type="checkbox"/> FSI <input type="checkbox"/> FTV	Année académique / Studienjahr 2008/09	No TD / Nr. DA it/2009/12
Mandant / Auftraggeber <input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input checked="" type="checkbox"/> Etablissement partenaire Blekinge Institute of Technology, Karlskrona	Etudiant / Student Carlo Arnold	Lieu d'exécution / Ausführungsort <input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input checked="" type="checkbox"/> Etablissement partenaire
Professeur / Dozent François Corthay	Expert / Experte (données complètes) Patrik Arlos	
Travail confidentiel / vertrauliche Arbeit <input type="checkbox"/> oui / ja <input checked="" type="checkbox"/> non / nein		

Titre / Titel

Ethernet traffic measurement point

Description et Objectifs / Beschreibung und Ziele

The aim of this project is to continue a previous diploma work on an Ethernet Measurement Point (MP) for a Distributed Passive Measurements Infrastructure (DPMI). In that diploma work, a sniffer board was realized and a MP circuit was developed on an Actel FPGA board.

In this diploma work:

- The MP hardware is to be transferred to a ML405 Xilinx FPGA board
- The Ethernet packet filters have to be further developed
- The system should also comply with the DPMI API v0.6 and/or v0.7.

Signature ou visa / Unterschrift oder Visum Resp. de la filière Leiter des Studieng.: Etudiant / Student: Carlo Arnold	Délais / Termine Attribution du thème / Ausgabe des Auftrags: 18.02.2009 Remise du rapport / Abgabe des Schlussberichts: août / August 2009 Exposition publique / Ausstellung Diplomarbeiten: 04.09.2009 Défense orale / Mündliche Verfechtung: à convenir / nach Vereinbarung
--	--

2010/DE/NEBIS 6000443
Hes-so VALAIS WALLIS

Rte du Rawyl 47 - 1950 Sion 2

- 4 SEP. 2009

Rapport reçu le / Schlussbericht erhalten am Visa du secrétariat / Visum des Sekretariats

System engineering - Infotronics

Bachelor thesis in the subject of

Ethernet traffic measurement point

September 3, 2009

Author: Carlo Arnold
Under guidance of: François Corthay (HES-SO) and Patrik Arlos (BTH)
Version: 5.0

Karlskrona, 18 August 2009

Contents

1	Preface	1
1.1	Introduction	1
1.2	Appendixes	1
1.3	Glossary	1
2	Overview	1
3	Material	2
3.1	Introduction	2
3.2	Interface board	3
3.3	Converter Board	4
3.3.1	Introduction	4
3.3.2	Pin assignment	5
3.3.3	Physical connection	5
3.3.4	Revisions	6
3.3.5	JTAG chain	6
3.4	Controller board (ML405)	7
3.5	Software development tools	8
4	Specification	9
4.1	Introduction	9
4.2	DPMI	9
4.2.1	Introduction	9
4.2.2	Ethernet frame structure	10
4.2.3	DPMI messages	12
4.2.4	Filter	16
4.3	TDS	17
4.3.1	Introduction	17
4.3.2	RS-422	18
4.3.3	Signals	18
4.4	Ethernet frames	18
4.5	DHCP	19
4.5.1	Introduction	19
4.5.2	DHCP procedure	19

4.6	ARP	21
4.6.1	Introduction	21
4.6.2	ARP announcement	21
4.6.3	ARP request handling (ARP reply)	22
5	VHDL development	22
5.1	Introduction	22
5.2	Basic structure	24
5.3	The top level	27
5.4	Capture interface	27
5.5	Filter database (FDB)	28
5.6	Arbiter	29
5.7	CI Demux	29
5.8	MArN interface	29
5.8.1	Introduction	29
5.8.2	Functionality	30
5.8.3	Start-up procedure	31
5.8.4	Receiver controller	33
5.8.5	State of development	34
5.9	Time synchronization client (TSC)	34
5.9.1	Introduction	34
5.9.2	Clock drift correction	35
5.9.3	Delay calculation	35
5.9.4	Time decoder	35
5.9.5	Picoseconds counter	39
5.10	Reset synchronization	40
6	Own TDS (GPS board)	40
6.1	Introduction	40
6.2	RS-422 interface	40
6.3	Physical properties	41
6.4	Realization	42
6.5	Progress and stage of development	43
7	Conclusion and remarks	45
8	Appendixes	47
9	Glossary	48

List of Figures

1	Basic concept of a Measurement Point	2
2	Boards	3
3	Interface board	4
4	JTAG chain represented by Xilinx iMPACT	6
5	Xilinx ML405 Evaluation Platform	7
6	Example of a MAr	9
7	Measurement frame	10
8	Measurement header	11
9	Capture header	11
10	Timestamp	12
11	Typical Time Server Distribution System	17
12	Typical DHCP session	19
13	ARP packet structure	21
14	Programmable logic arrays	22
15	Xilinx Parallel Cable IV	23
16	Xilinx CORE Generator Wizard	25
17	Xilinx Simulation Library Compilation Wizard	26
18	Schematic of the topLevel of the FPGA design	27
19	Schematic of the CI	28
20	RAM structure FDB	29
21	Schematic of the MArN interface	30
22	Start-up procedure	32
23	Graph of the statemachine of the receiver_ctrl block	33
24	Schematic of the TSC client	34
25	Graph of the statemachine of the delay_calc block	35
26	Schematic of the time_decoder block	36
27	Principle graph of the statemachine of the NMEA_decoder block	37
28	“date” command to convert UNIX timestamp	38
29	Wave diagram of ModelSim simulation of the time decoder	39
30	GPS board	42
31	Schematic of the FPGA design	43
32	PPS on the oscilloscope	44

List of Tables

1	Unconnected signals of the Interface board	5
2	MAINFO message content	12
3	MPInitialization message content	13
4	MPStatus message content	13
5	MPFilter (new filter) message content	14
6	MPFilter (change filter) message content	14
7	MPVerify message content	14
8	Filter content	16
9	Used EtherType values	18
10	HDL Designer Libraries	24
11	Codes for the transmit_mode signal	31
12	Codes for the static_msg_id signal	31
13	TDS signals (masters view)	41

1 Preface

1.1 Introduction

To finalize the degree program of system engineering at HES-SO Valais in Sion, Switzerland the students get the opportunity to do their bachelor thesis in a foreign country. I took this chance and obtained the possibility to write my thesis at BTH in Karlskrona, Sweden under the supervision of Patrik Arlos on a very interesting topic called “Ethernet traffic measurement point”.

This bachelor thesis is a continuation of a previous diploma work of Silvan Zahno about an Ethernet Measurement Point (MP) for a Distributed Passive Measurement Infrastructure (DPMI). In that previous diploma work a sniffer board was realized and an MP circuit was developed on an Actel CoreMP7 FPGA board. The goal of this thesis is to transfer the MP hardware to a Xilinx ML405 FPGA Board and to improve the Ethernet packet filter functions. The necessary hardware for this project was designed and developed at HES-SO in Sion, Switzerland. All VHDL development was done at BTH in Karlskrona, Sweden.

The goal of this document is to explain some theoretical details but also to document the work that was done during the 12 weeks in Karlskrona and the preparation time in Sion.

1.2 Appendixes

If there is some useful information regarding a subject of this documentation it will be attached as an appendix. There will be a reference in the text if there is an appendix available for this topic. For a list of all appendixes please see the chapter 8.

1.3 Glossary

To make it easier to read this documentation, abbreviations are used. The first time a term is mentioned its abbreviations is explained. As a help there is a glossary at the end of this document in chapter 9 in which all abrogations are explained again.

2 Overview

The Ethernet traffic measurement point (MP) is a hardware equipment that captures Ethernet frames, filters them and adds a timestamp on each packet. This data is stored in a buffer. As soon as this buffer reaches a certain threshold, the data will be sent to the Measurement Area Network (MArN). The timestamp will be deployed by a Time Distributing Server (TDS), which gets the exact time over a GPS radio signal. The FPGA in the MP is responsible to add the correct timestamp on the packets, to filter them and to send this information to the MArN. The MArN and the MP are part of a Distributed Passive

Measurement Infrastructure (DPMI) which is a concept developed by Patrik Arlos at BTH. This infrastructure is able to analyze the captured network data and also to control the MP. The following figure explains this system.

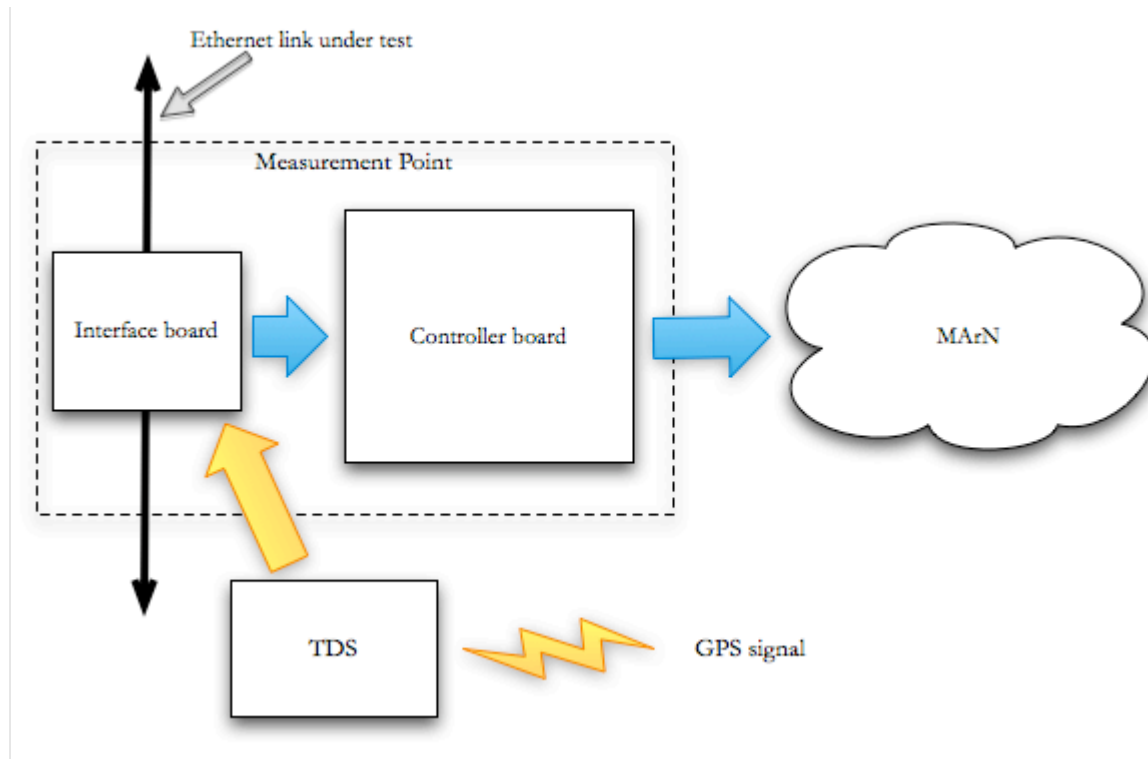


Figure 1: Basic concept of a Measurement Point

3 Material

3.1 Introduction

The hardware of the MP is built of three boards. The Interface, the Converter and the Controller board. The Converter board connects the other two boards as shown on the following figure.

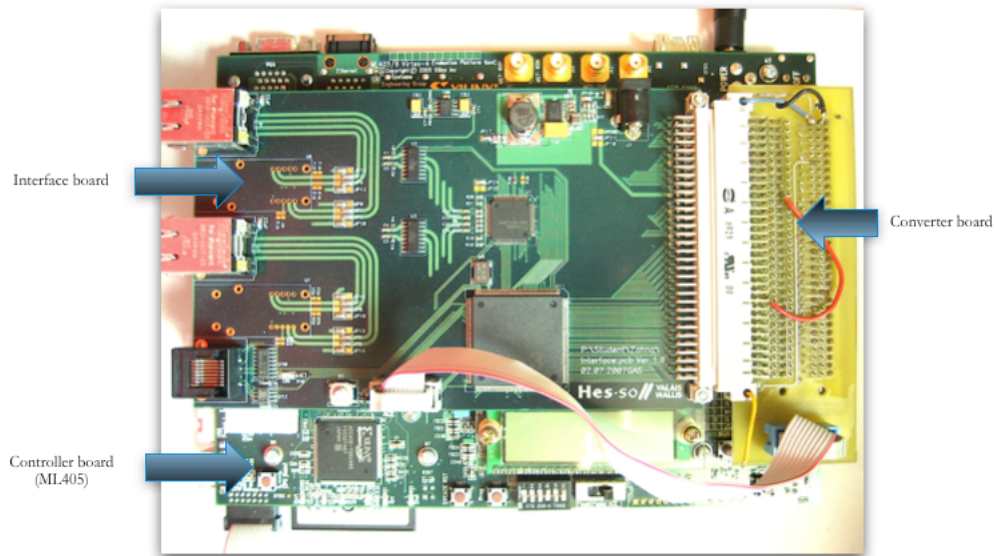


Figure 2: Boards

This chapter describes these three boards.

3.2 Interface board

As previously mentioned, the sniffer board that captures the Ethernet frames was already developed by Silvan Zahno, a former exchange student from Switzerland at BTH in Karlskrona. This board is called Interface board.

The Interface board simply sniffs on the Ethernet link that is being tested and sends the packets through a Complex Programmable Logic Device (CPLD) to the Controller board, which in our case is the Xilinx ML405 FPGA board. To operate at a lower frequency the Interface board scales the data up to 32 bits per port. The Interface port sniffs passively, which means that it cannot be recognised from other network devices.

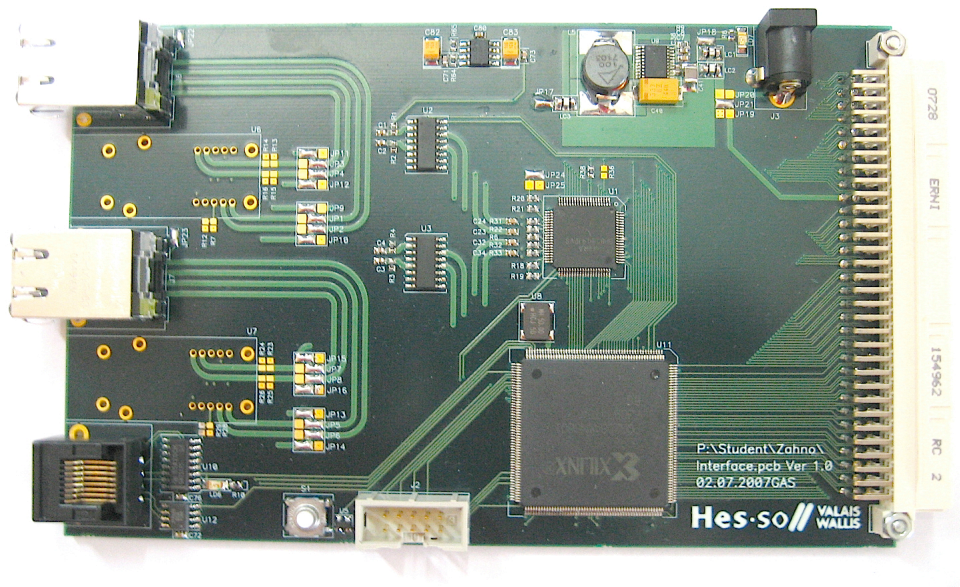


Figure 3: Interface board

Basically, the Interface board has 4 important connectors. Two RJ-45 Ethernet ports to insert the board into the Ethernet link to test, one additional RJ-45 port to connect a TDS over RS-422 and one 96 pin connector to attach the Controller board. There also is a 10 pin connector for JTAG which makes it possible to program the CPLD on the board. Because the receiving (RX) channel of one Ethernet port is the transmitting (TX) port of the other, it is sufficient to listen to the RX channels of both ports.

For further information about the Interface board, please see the chapter 4.1 of Silvan Zahno's diploma work report (appendix 1).

3.3 Converter Board

3.3.1 Introduction

To connect the Interface board to the Xilinx ML405 FPGA board, I had to build a Converter board. First I had to verify if the ML405 board has enough I/Os and is able to supply the Interface board with power. The Interface board has a 96pin connector which contains all signals and the power lines. The ML405 has an expansion header with 32 single-ended I/Os, 16 differential pairs (which can be used as another 32 single-ended I/Os), 14 spare I/Os shared with push-button switches and LEDs, power (5V and 3.3V) and a JTAG chain expansion capability. There are also 2 pins on the expansion header for Inter-Integrated Circuit (IIC or I2C) bus expansion which are not used for this project. So in total there

are 78 I/Os that can be used and the required voltages of 3.3V and 5V are available.

3.3.2 Pin assignment

The Interface board has a total of 88 signals that are on the 96 pin connector. Five pins are for reserve and three for power. So there are 10 additional signals then the amount of I/O ports of the ML405. Fortunately there are some control signals that are not required for a properly working MP. The following signals of the Interface board are not connected to the ML405.

Signal name	Description
RXEN_OUT	To disable high impedance buffer for the Ethernet transceiver (RX) (can be set to 1 by the CPLD)
TXEN_OUT	To disable high impedance buffer for the Ethernet transceiver (TX) (can be set to 1 by the CPLD)
MDIO_OUT	Serial management interface of the Ethernet transceiver (not used)
MDC_OUT	Serial management interface of the Ethernet transceiver (not used)
TX_CLK_A_OUT	Ethernet clock of port A
TX_EN_A_OUT	Ethernet enable of port A
PWRDOWN_INT_A_OUT	Power down interrupt for port A
TX_CLK_B_OUT	Ethernet clock of port B
TX_EN_B_OUT	Ethernet enable of port B
PWRDOWN_INT_B_OUT	Power down interrupt for port B

Table 1: Unconnected signals of the Interface board

By not connecting these signals there are 78 signals left and that is exactly the amount of I/O ports available on the ML405. All the other data- and control-signals have been kept.

3.3.3 Physical connection

The Converter board consists of three connectors. One 96 pin connector to attach the interface board and two 96 pin connectors for the expansion header on the ML405. To mount the MP into a case it is advantageous to have the front of the Interface board on the same geometrical lining-up as the ML405. So the Converter board was designed to achieve this specification. See the figure 2 for a better understanding.

3.3.4 Revisions

In the first revision of the Converter board I unfortunately made a design fault relating to the power- and ground-lines. That is why a second revision has been designed. The electrical schematic and the PCB layout of the second revision can be found in the appendixes 2 and 3.

3.3.5 JTAG chain

Because the expansion header of the ML405 is equipped with a Joint Test Action Group (JTAG) chain expansion it comes in handy to place a JTAG connector on the converter board. This makes it possible to connect the JTAG interface of the Interface board with the ML405 through the Converter board. So it is possible to program the CPLD on the Interface board over the same JTAG chain on which the FPGA and the CPLD of the ML405 is connected. That results in the fact that a PC can be connected to the JTAG connector on the ML405 and from there every device in the whole MP can be programmed. Thinking again of case-mounting, it would be possible to have a JTAG connector on the front plate of the case, so that it would not be necessary to open the case to reprogram any device.

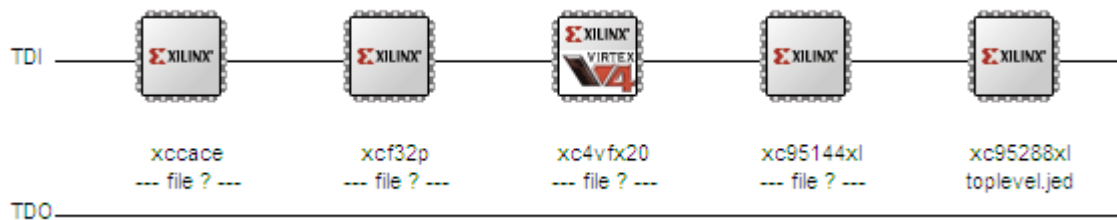


Figure 4: JTAG chain represented by Xilinx iMPACT

The devices in the JTAG chain consist of:

- xccace: System ACE Controller (Compact Flash (CF) configuration controller)
- xcf32p: Platform Flash
- xc4vfx20: Virtex-4 FPGA on ML405
- xc95144xl: CPLD on ML405
- xc95288xl: CPLD on the Interface board

To enable the JTAG chain extension the jumper J26 on the ML405 has to be set correctly. (see appendix 4, page 26)

3.4 Controller board (ML405)

The Controller board of the MP will be realized with a Xilinx ML405 board¹. It is equipped with a Virtex-4 FX FPGA (XC4VFX20-FF672).

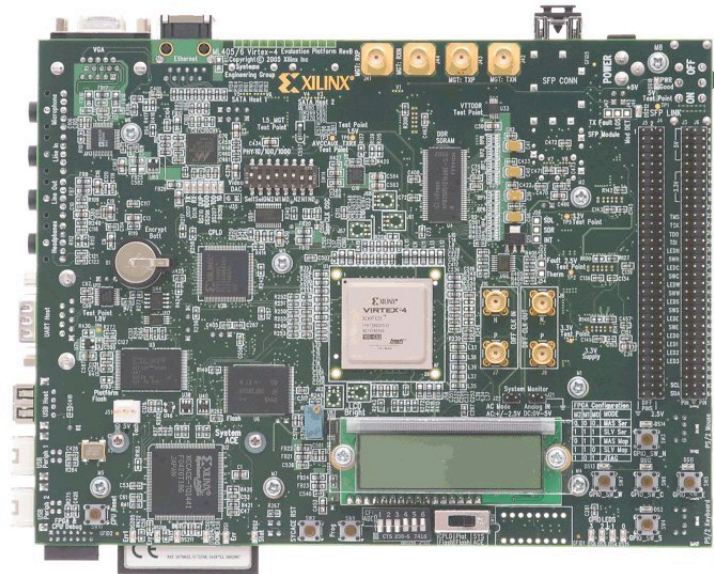


Figure 5: Xilinx ML405 Evaluation Platform

Key features:

- 64-Mbyte DDR SDRAM
- 10/100/1000 tri-speed Ethernet PHY transceiver
- USB interface chip (Cypress CY7C67300) with host and peripheral ports
- Expansion header with 32 single-ended I/O, 16 LVDS capable differential pairs, 14 spare I/O's shared with buttons and LEDs, power, JTAG chain expansion capability, and IIC bus expansion
- Stereo AC97 audio
- RS-232 serial port
- 16-character x 2-line LCD display
- 4-kb IIC EEPROM
- VGA output with 50 MHz / 24-bit video DAC

¹<http://www.xilinx.com/support/documentation/ml405.htm>

- PS/2 mouse and keyboard connectors
- System ACE™ CompactFlash configuration controller with Type I/II CompactFlash connector
- ZBT synchronous SRAM (9 Mb) on 32-bit data bus with four parity bits
- Intel StrataFlash (or compatible) linear flash memory chips (8 Mbyte)
- Xilinx XC95144XL CPLD for FPGA configuration
- Two SATA connectors
- One SFP connector
- Four SMA connectors accessing one channel of RocketIO Transceiver

The ML405 satisfies all our needs. Besides the powerful FPGA and a big SDRAM there is the expansion header that allows us to connect the Interface board through the Conversion board. Furthermore an Ethernet PHY transceiver is available to connect the MP to the MarN. The LCD display could be useful for debug applications.

3.5 Software development tools

The Xilinx Virtex-4 FPGA on the ML405 Evaluation Platform will be programmed in VHDL. This abbreviation stands for Very High Speed Integrated Circuit Hardware Description Language which is a programming language that describes logical circuits (hardware). To develop the VHDL design a couple of software tools are necessary. The following tools have been used for this project.

- Mentor Graphics HDL Designer 2007.1a
- Mentor Graphics ModelSim SE 6.3g
- Xilinx ISE 9.2i

The Mentor Graphics HDL Designer² is used to develop the VHDL design. It is possible to do most of the development in a graphical way. It has been used because it is compatible to other FPGAs from other manufacturers.

ModelSim SE allows to simulate the VHDL design. This is very useful to detect design faults before loading the design on the FPGA.

ISE is a large design tool from Xilinx³ and since the FPGA for this project is manufactured by Xilinx, ISE is the ideal solution to program it. The VHDL code that was created with the Mentor Graphics HDL Designer will be the input for ISE. The synthesis, place & route and programming processes will be handled by the ISE tools.

The development workstation is a PC with Windows XP Professional Version 2002 and SP3.

²<http://www.mentor.com/>

³<http://www.xilinx.com/>

4 Specification

4.1 Introduction

Before it is possible to develop the MP it is necessary to know the theoretical functioning of the MP itself and the environment in which it will be operating. This chapter explains the technologies that are used by the MP.

4.2 DPMI

4.2.1 Introduction

If the MP is able to communicate on the MArN over UDP/IP and ARP (see chapters 4.5 and 4.6) it has to connect to the Measurement Area Controller (MArC). This concept is defined as DPMI. This chapter will explain this concept and some important information in a detailed manner. According to its name, the MArC controls the MPs and consumers that are connected to the MArN. The following figure shows an example of a Measurement Area (MAr) of a DPMI.

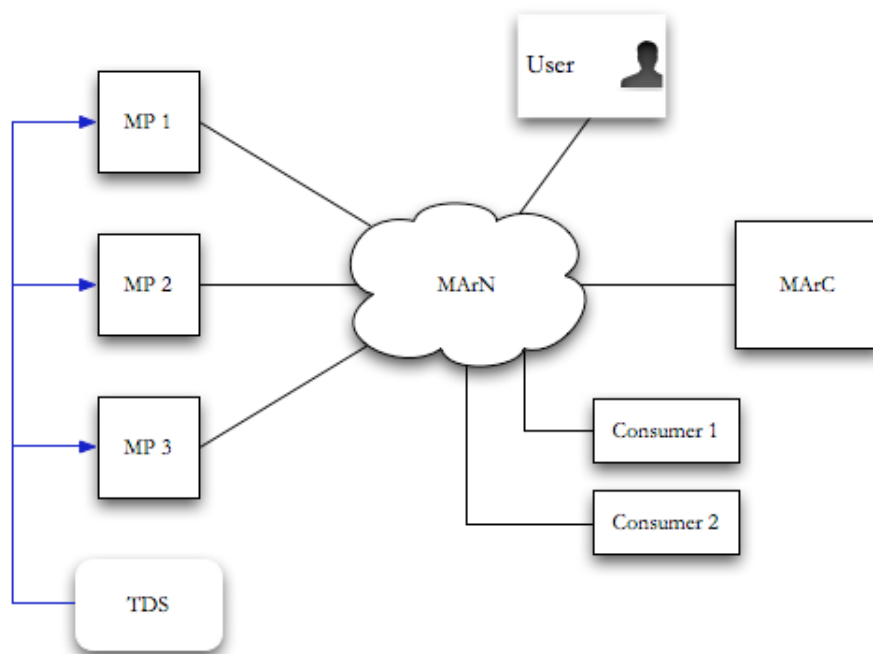


Figure 6: Example of a MAr

This example infrastructure shows a MAr with three MPs and two consumers. Each MP measures the traffic of another link which can be of any imaginable physical form. Every MP gets a time signal from a Time Distributing Server (TDS). More information about the TDS follows in chapter 4.3. The MPs get their configurations from the MArC, the MArC itself is user-managable over a web interface. This web interface could be provided by another computer, not necessarily the one the MArC runs on, but as this is not relevant for the understanding of the MP, the MArC is considered as one logical element. Once an MP is authorised to capture data, it will send the measurement frames as UDP packets to the MArN. These measurement frames are sent using Ethernet multicast. This makes it possible that multiple consumers are gathering data from the same MP. Once the captured data reaches a consumer it can be treated there or transferred through a secure channel like a VPN tunnel to another network. Inside the MArN the data is not encrypted. For further information about the DPMI please see appendix 5.

4.2.2 Ethernet frame structure

4.2.2.1 Introduction To generate correctly formed Ethernet frames it is necessary to know how the frames are defined in the DPMI standard. While analyzing a header file of a software MP developed by Patrik Arlos (see appendix 6) it was possible to work this out. This chapter will explain the packet structure and its purpose.

4.2.2.2 Measurement frame The following figure shows a measurement frame as it looks like on the Ethernet. The numbers underneath the several elements indicate the number of bytes that are used for this element.

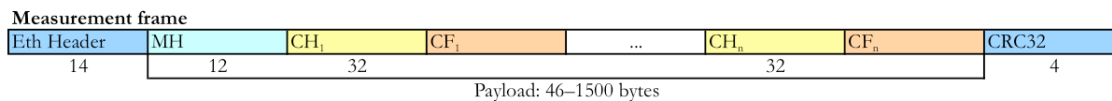


Figure 7: Measurement frame

The measurement begins with a usual Ethernet header and ends with a check-sum (CRC). This is given by the IEEE 802.3 standard. The payload in between can be freely used. It just has to be between 46 and 1500 bytes long. For more information about Ethernet frames please see chapter 4.4. Each measurement frame has got a measurement header (MH). Then it can contain multiple captured frames (CF) that always have a corresponding capture header (CH) describing them.

Remark: As the measurement frames are sent using Ethernet multicast the destination MAC address in the Ethernet header has to be adjusted. Therefor the MArN has to be

analyzed. It is also possible to send these frames to the broadcast (BC) address or to the MAC address of a specific consumer. In every case it is the consumer which decides how it wants to treat the arriving packets.

4.2.2.3 Measurement header (MH) The 12 bytes that form the MH are arranged as followed.

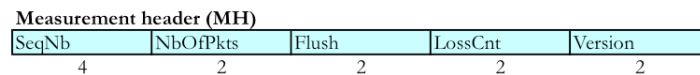


Figure 8: Measurement header

- Sequence number: Indicates the number of sent data packets. So after a packet is sent this value is incremented by one.
- Number of packets: Indicates the number of CF in this measurement frame.
- Flush: Indicates that this is the last packet.
- Loss counter: Indicates how many protocol data units (PDU) were lost during the creation of this frame.
- Version: Indicates what version of the file format is used for storing the packets.

4.2.2.4 Capture header (CH) The capture header is composed as followed and is 32 bytes long.

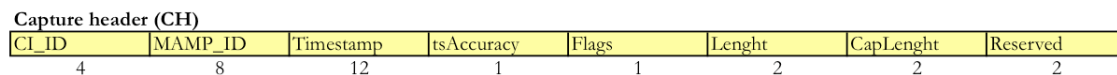


Figure 9: Capture header

- Capture Interface id: Identifies the CI where the frame was captured.
- MAMP id: Identifies the MP where the frame was captured.
- timestamp: Identifies when the frame was caught. A more detailed description of this element is in the next chapter.
- Timestamp accuracy: Indicates the accuracy of the timestamp so the number of digits to trust.
- Flags: Indicates the flags set for the PDU.
- Length: Indicates the length of the frame.
- Captured length: Indicates how much of the captured frame can be found here.

- Reserved: This element contains various flags. It is needed to make the header a multiple of 32 bits.

4.2.2.5 Timestamp This figure shows a detailed view of the 12 bytes that are used for the timestamp.

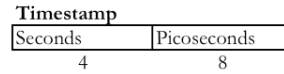


Figure 10: Timestamp

The first 4 bytes contain the number of seconds since 00:00:00 UTC on January 1, 1970. This is the UNIX timestamp. The other 8 bytes store the number of picoseconds between the start of the current second and the moment when the frame was captured. It is actually a fractal part of a second.

4.2.3 DPMI messages

4.2.3.1 Introduction To control the MP messages are exchanged between the Measurement Area Controller (MArC) and the MP. The following chapters describe those messages. They are transmitted in UDP/IP packages. The number in brackets in the chapter titles indicates the message ID.

4.2.3.2 Initializing message (MAINFO) This message called MAINFO is special as it has no message ID. It is sent by the MArC if it receives an empty UDP package on the port 1500. The following table shows what information this message contains.

Value	Size	Description
version	32 bits	DPMI version (1 corresponds to version 0.5 and 2 to version 0.6)
address	16 bytes	IP address of the MArC (coded in ASCII)
port	32 bits	UDP port of the MArC
database	64 bytes	not used in DPMI version 0.6
user	64 bytes	not used in DPMI version 0.6
password	64 bytes	not used in DPMI version 0.6

Table 2: MAINFO message content

4.2.3.3 Authorization message (1) This message is called MPInitialization and means that the MP is authorised and can start sending captured data on the MArN. It contains the following information.

Value	Size	Description
type	32 bits	Message ID is always 1
mac	8 bytes	MAC address of the MP
name 200 chars	200 bytes	Name of the MP (coded in ASCII)
ipAddress	32 bits	IP address of the MP
port	16 bits	UDP port of the MP
maxFilters	16 bits	Number of filters the MP can handle
noCI	16 bits	Number of CIs of the MP
MAMPid	16 bytes	MP's ID which is set by the MArC

Table 3: MPInitialization message content

Before the MP gets this message it has to send the same type of message with its information. The MArC will sent back this message but insert the MP's ID (MAMPid) in the message.

4.2.3.4 Status message (2) This message is called MPStatus and is sent by the MP every second to inform the MArC about its status. It also is used for statistical purposes.

Value	Size	Description
type	32 bits	Message ID is always 2
MAMPid	16 bytes	MP's ID which is set by the MArC
noFilters	32 bits	Number of active filters in the MP
matched	32 bits	Number of packages that matched one of the filters
noCI	32 bits	Number of CIs of the MP
CIstats	1100 bytes	ASCII string that contains some more statistic information

Table 4: MPStatus message content

The field CIstats contains an ASCII string in the following format.

```
CI1:Recv.Packets:Filtered.Packets:BufferUse
CI2:Recv.Packets:Filtered.Packets:BufferUse
```

4.2.3.5 New filter indication message (3) This message is called MPFilter and indicates that the MP has to add a new filter.

Value	Size	Description
type	32 bits	Message ID is always 3
MAMPid	16 bytes	MP's ID which is set by the MArC
theFilter	168 bytes	New filter

Table 5: MPFilter (new filter) message content

4.2.3.6 Change filter indication message (4) Since every filter has its proper index number it is possible to change a filter for a given index number. That is the reason for this message. It is actually the same message structure as the new filter indication message (MPFilter), but with the message ID 4 it is indicated, that it is just a changed filter and not a new one.

Value	Size	Description
type	32 bits	Message ID is always 4
MAMPid	16 bytes	MP's ID which is set by the MArC
theFilter	168 bytes	Changed filter

Table 6: MPFilter (change filter) message content

The field theFilter contains a filter_id. So the MP knows which filter it has to change.

4.2.3.7 Drop filter indication message (5) This message simply indicates that a filter should be deleted. But as there is no message structure available this feature is not used.

4.2.3.8 Filter verification message (6) When the MP receives this message called MPVerify it has to send the filter with the index number contained in the message back to the MArC. So it is possible to verify it. The MP sends back the same message type but with the filter it has stored.

Value	Size	Description
type	32 bits	Message ID is always 5
MAMPid	16 bytes	MP's ID which is set by the MArC
theFilter	168 bytes	Filter to check

Table 7: MPVerify message content

4.2.3.9 All filters verification message (7) This message requests the MP to send all filters to the MArC for verification. It is not used.

4.2.3.10 Termination message (8) This message requests the MP to shut down. For security reasons it must contain a specific magic word. This will not be implemented as it is not needed to shutdown our MP remotely.

4.2.3.11 Flush buffers request message (9) If this message arrives the MP has to flush its buffers. This can be requested by the MArC if it wants to be sure that there is no data of a previous measurement left in the buffers. There even would be a possibility to flush a specific buffer but this is not implemented in this MP.

4.2.4 Filter

The filters that are exchanged between the MP and the MArC have a certain format. Please see the following table to see its fields.

Value	Size	Description
filter_id	32 bits	Filter ID
index	32 bits	Specifies which fields have to be checked. Each bit represents a field.
CI_ID	8 bytes	CI name
VLAN_TCI	16 bits	VLAN ID
ETH_TYPE	16 bits	Ethertype
ETH_SRC	6 bytes	Ethernet source address
ETH_DST	6 bytes	Ethernet destination address
IP_PROTO	8 bits	IP protocol field (TCP or UDP)
IP_SRC	16 bytes	IP source address (coded in ASCII)
IP_DST	16 bytes	IP destination address (coded in ASCII)
SRC_PORT	16 bits	Source port
DST_PORT	16 bits	Destination port
VLAN_TCI_MASK	16 bits	VLAN ID bit mask
ETH_TYPE_MASK	16 bits	Ethertype bit mask
ETH_SRC_MASK	6 bytes	Source Ethernet address bit mask
ETH_DST_MASK	6 bytes	Destination Ethernet address bit mask
IP_PROTO_MASK	8 bits	IP protocol field bit mask
IP_SRC_MASK	16 bytes	IP source address bit mask (coded in ASCII)
IP_DST_MASK	16 bytes	IP destination address bit mask (coded in ASCII)
SRC_PORT_MASK	16 bits	Source port bit mask
DST_PORT_MASK	16 bits	Destination port bit mask
DESTADDR	22 bytes	Destination address
DESTPORT	32 bits	Destination port
TYPE	32 bits	Consumer stream type

Table 8: Filter content

The different values and bit masks are applied on the headers of captured data frames. The filter_id field contains a unique ID for each filter of an MP and the index field is a 32 bit field that specifies which of the following 22 fields should be checked.

4.3 TDS

4.3.1 Introduction

As already mentioned, a Time Distributing Server (TDS) is used to synchronize all devices in the MAR. There are commercial solutions like the TDS-2 from Endace⁴. Please see the following figure from the TDS-2 documentation (appendix 7) and the explanations to understand how a TDS distributes a time signal.

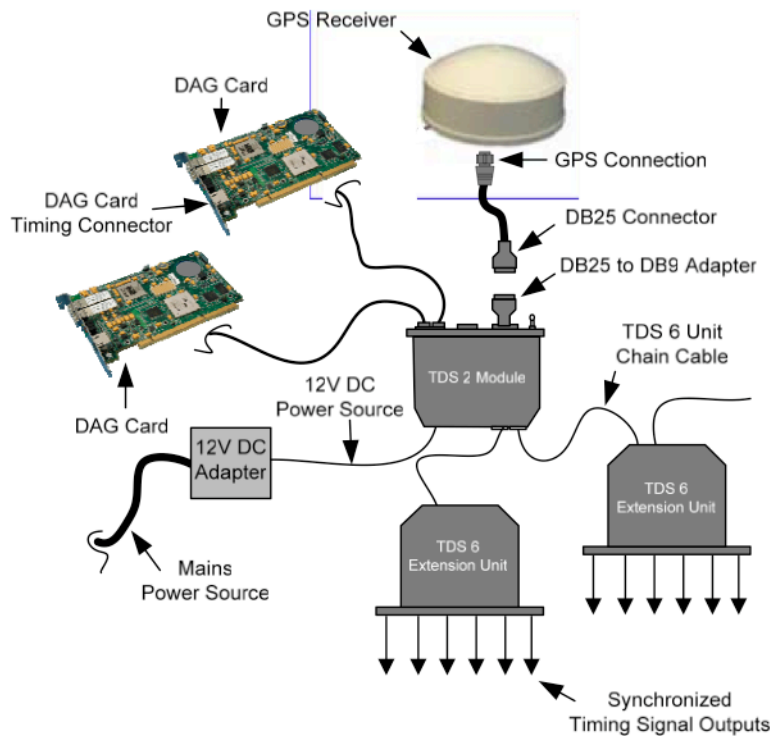


Figure 11: Typical Time Server Distribution System

Basically the TDS-2 receives a time signal from a GPS antenna and distributes it over RS-422 lines to DAG cards⁵. DAG stands for Data Acquisition and Generation. These cards are commercial products from Endace that could be replaced by the MP which is developed in this thesis. The TDS-6 that is shown in the figure above is an extension unit that makes it possible to supply additional devices with the time signal.

⁴<http://www.endace.com/>

⁵<http://www.endace.com/dag-network-monitoring-cards.html>

4.3.2 RS-422

The advantage of the RS-422 standard compared to the RS-232 is that it is using differential signaling, which makes it possible to transmit data over much longer distances. This means that for a single signal two electrical lines are used. Between the transmitter and the receiver is one positive and one negative line. RS-422, also known as EIA-422, does not define any protocol that has to be used nor the connectors are defined. In this project usual RJ45 connectors are used.

4.3.3 Signals

There are two signals needed from a TDS to have a proper time synchronization. One is the Pulse Per Second (PPS) and the other one is a serial asynchronous time and date signal. The PPS is a signal that is provided by the GPS. It makes it possible to determine very precisely the moment when a second is starting. The time and date signal is also provided by the GPS and coded in the NMEA 0183 protocol⁶. This protocol was developed by the National Marine Electronics Association⁷ and it is used by nearly every GPS receiver.

4.4 Ethernet frames

To generate correct Ethernet frames it is important to know how an Ethernet frame looks like in general. Every packet sent by the MP must be sent using the IEEE 802.3 standard. Basically an Ethernet frame has a header, a payload and a CRC checksum in the trailer. The header of an Ethernet frame is 14 bytes long. It contains the destination MAC address, the source MAC address and a two bytes long field called EtherType. The EtherType indicates which protocol is encapsulated in the payload. We use the following EtherType values.

EtherType	Protocol
0x0800	Internet Protocol (IP)
0x0806	Address Resolution Protocol (ARP)
0x0810	DPMI measurement frames

Table 9: Used EtherType values

The EtherType for the measurement frames is defined in DPMI. The value 0x0810 is not used by any other protocol so it is possible to use it. The user datagram protocol (UDP) runs over IP. This means all DHCP communication works over UDP/IP.

⁶<http://www.gpsinformation.org/dale/nmea.htm>

⁷<http://www.nmea.org/>

4.5 DHCP

4.5.1 Introduction

Because the MP communicates with the MArC over the UDP/IP protocol, a DHCP client has to be implemented. DHCP stands for Dynamic Host Configuration Protocol and is a protocol that is used by network devices to obtain configuration information as the IP address and the subnet mask from a DHCP server. With a DHCP client the MP can be connected to any MArN with a DHCP server and connect automatically to the MArC.

4.5.2 DHCP procedure

The following figure illustrates a typical DHCP session. For the conversation several messages⁸ are transmitted.

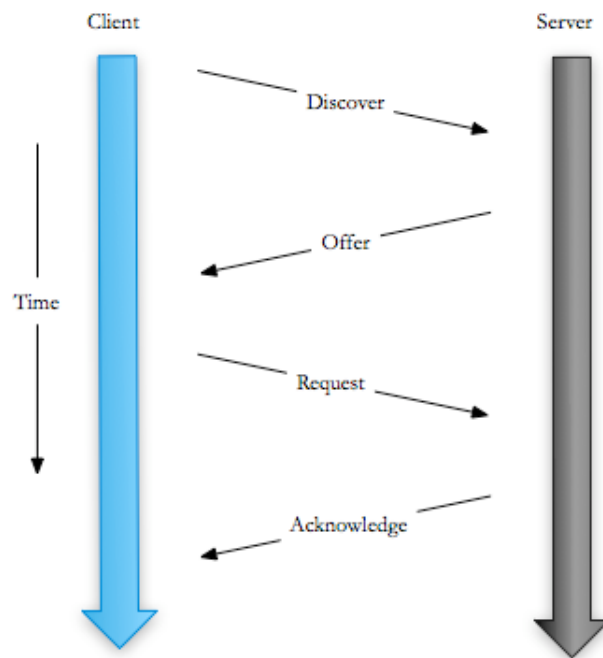


Figure 12: Typical DHCP session

This chapter will describe these four messages (DHCPDISCOVER, DHCPOFFER, DHCPREQUEST and DHCPACK). It is important to understand its function to understand how the DHCP client is implemented. All these messages are exchanged as UDP packets. On

⁸<http://support.microsoft.com/kb/169289>

the server site the UDP port 67 is used and on the client site it is port 68. The DHCP message type is coded in the field DHCP option 53 in a DHCP packet.

4.5.2.1 DHCP Discover This message is sent by the client as a broadcast (BC) on the physical line. As the client has no IP address yet, it sends this message with the source address 0.0.0.0 and the destination address 255.255.255.255 which is the BC address. It would be possible that the client saves its last obtained IP address and put it as DHCP option 50 in this message. This will not be used in our project. The client puts its hardware address (MAC address) in the field CHADDR (Client Hardware Address) and sends the packet.

4.5.2.2 DHCP Offer Like all of these four messages, the offer will also be sent to the BC address. This time the source address will be the DHCP server's IP address. The server chooses a free IP address and puts it in the field YIADDR (Your IP Address) and in the CHADDR field it will leave the MAC address. In the DHCP option 3 the server will place its IP address. It also will add the subnet mask in option 1, the IP addresses of the DNS servers in option 6 and the IP lease time in option 51. In our case the DNS server are not used but the IP lease time is important. It has to be saved and after passing half of the number of seconds the lease time indicates the client must to do another DHCP request.

4.5.2.3 DHCP Request As there could be multiple DHCP servers making an offer the client now has to make a DHCP request to the server he accepts. Since there is only one DHCP server present in a MArN, the MP will sent the request after the first offer arrived. In the DHCP request the client will put the IP address that was received in the offer into the DHCP option 50 and sent it back to the server. The first time a request is send it will be sent as a BC message. This allows potential additional DHCP servers in the network to know that another DHCP server has been chosen. A request will also be sent after half of the lease time. In that case it will be sent as a unicast directly to the DHCP server. This means the destination address of this packet is the IP address of the DHCP server.

4.5.2.4 DHCP Acknowledge Now the server simply has to confirm to the client that it can use the specific IP. That is done with the DHCP Acknowledge message.

4.5.2.5 Other DHCP messages There are the following additional messages defined in the DHCP protocol. DHCPNAK, DHCPDECLINE, DHCPRELEASE and DHCPINFORM. For the MP only the NAK message is important. It would be sent by the server to refuse a request. That means for the MP if it receives a DHCPNAK it has to resend a DHCPREQUEST or even a DHCPDISCOVER after some attempts. If a termination

procedure would be implemented in the MP it would be nice if it would send a DHCPRELEASE message before terminating. If the DHCP server receives such a message it can use the IP address for another client. It is not critical if the release message is not sent as after the lease time is over the IP address will be released automatically by the DHCP server.

4.6 ARP

4.6.1 Introduction

ARP stands for Address Resolution Protocol. This protocol is used to make the link between the IP- and the MAC address of a network device. As the MArN is an Ethernet link it is necessary that the MP can handle ARP messages. This chapter describes how the MP has to support the ARP. On the following figure you can see how an ARP packet looks like.

bit offset	0 - 7	8 - 15	16 - 31
0	Hardware type (HTYPE)		Protocol type (PTYPE)
32	Hardware length (HLEN)	Protocol length (PLEN)	Operation (OPER)
64	Sender hardware address (SHA) (first 32 bits)		
96	Sender hardware address (SHA) (last 16 bits)		Sender protocol address (SPA) (first 16 bits)
128	Sender protocol address (SPA) (last 16 bits)		Target hardware address (THA) (first 16 bits)
160	Target hardware address (THA) (last 32 bits)		
192	Target protocol address (TPA)		

source: www.wikipedia.org

Figure 13: ARP packet structure

4.6.2 ARP announcement

If the MP has received an IP address from a DHCP server (see chapter 4.5) it has to communicate its IP- and MAC address to the other devices in the network, using an ARP announcement. Therefor the fields TPA and SPA has to be filled in with the identical destination and source addresses. (see figure 13) The SHA corresponds to the MAC address and the SPA and the TPA to the IP address. If this ARP announcement is sent there won't be any answer. It is just used to let other devices know that there is a new device in the network. Every device is now updating its ARP table. Especially for the MArC it is now possible to contact the MP over the IP. It is recommended that such an ARP announcement is sent every 15 minutes.

4.6.3 ARP request handling (ARP reply)

The MP has to be able to answer an ARP request. This means if a device wants to send something to the MP's IP address but it does not know the MP's MAC address it sends an ARP request. Such a request looks as follows. The SPA and the SHA correspond to the IP- and the MAC address of the requesting device. (see figure 13) The TPA is the IP address of the host the requester is looking for and the THA is the broadcast address ($FF-FF-FF-FF-FF-FF_{16}$). If the TPA corresponds to the IP address a reply must be sent. An ARP reply of the MP looks like this. The SPA and the SHA is the IP- and the MAC address of the MP. The TPA and the THA is the IP- and the MAC address of the host that made the request.

5 VHDL development

5.1 Introduction

The most important task of this project is to transfer an existing VHDL design and make it compatible with the ML405. Based on this design, additional features will be implemented. These are:

- Correct timestamping (in the correct format as understood by the DPMI consumers)
- DPMI compatible Ethernet interface (MArN)
- Time synchronization from TDS2 (Endace system)

This chapter describes the developed VHDL design in detail. The blocks that have been adopted from the previous design will be documented just briefly as they are already documented in Silvan Zahno's report (see appendix 1). To realize the MP, two programmable logic arrays are going to be programmed over a JTAG interface.

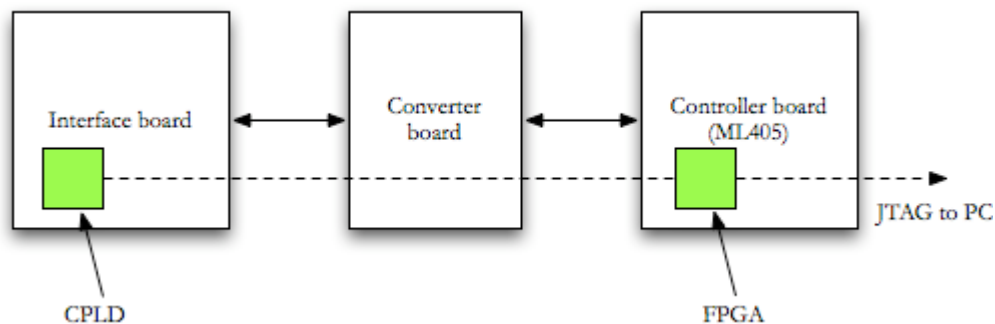


Figure 14: Programmable logic arrays

The CPLD is responsible to send the captured frames through the Converter board to the Controller board. There, the FPGA does all the MP functions. It filters the frames and creates the measurement frames that are sent to the MARN. As already explained in chapter 3 it is possible to program the CPLD (Xilinx XC95288XL) on the Interface board and the FPGA (Virtex-4 FX) using one JTAG chain. For the connection to the workstation PC the Xilinx Parallel Cable IV is used.



Figure 15: Xilinx Parallel Cable IV

5.2 Basic structure

The HDL Designer Project is called “mp” and contains multiple libraries. The following libraries has been created.

Library name	Description
mp_lib	This is the default library where all basic components are included.
components	This library contains all other components which are not imported from another library.
interface_Ethernet_lib	This library contains the design of a Ethernet interface. It was developed at HES-SO in Sion. For the documentation please see appendix 8.
xilinx_cores	This library contains the buffers that had been generated by the Xilinx CORE Generator.
XilinxCoreLib	This is a core library generated with the Xilinx Simulation Library Compilation Wizard and it is used by the buffer in the virtex4 library.
mp_lib_tb	This library contains a testbench for the topLevel block which is defined in the mp_lib library.

Table 10: HDL Designer Libraries

Remark: Buffers To implement a buffer the CORE Generator included in Xilinx ISE 9.2i has been used. First an empty library in the HDL Designer project has been prepared. Then the buffer core with the CORE Generator has been generated into the directory of that new library. By adding the VHDL source file of the new buffer core to the library the buffer is available in the project as a normal component. If one day the hardware (the FPGA) changes, the buffer core has to be changed as well.

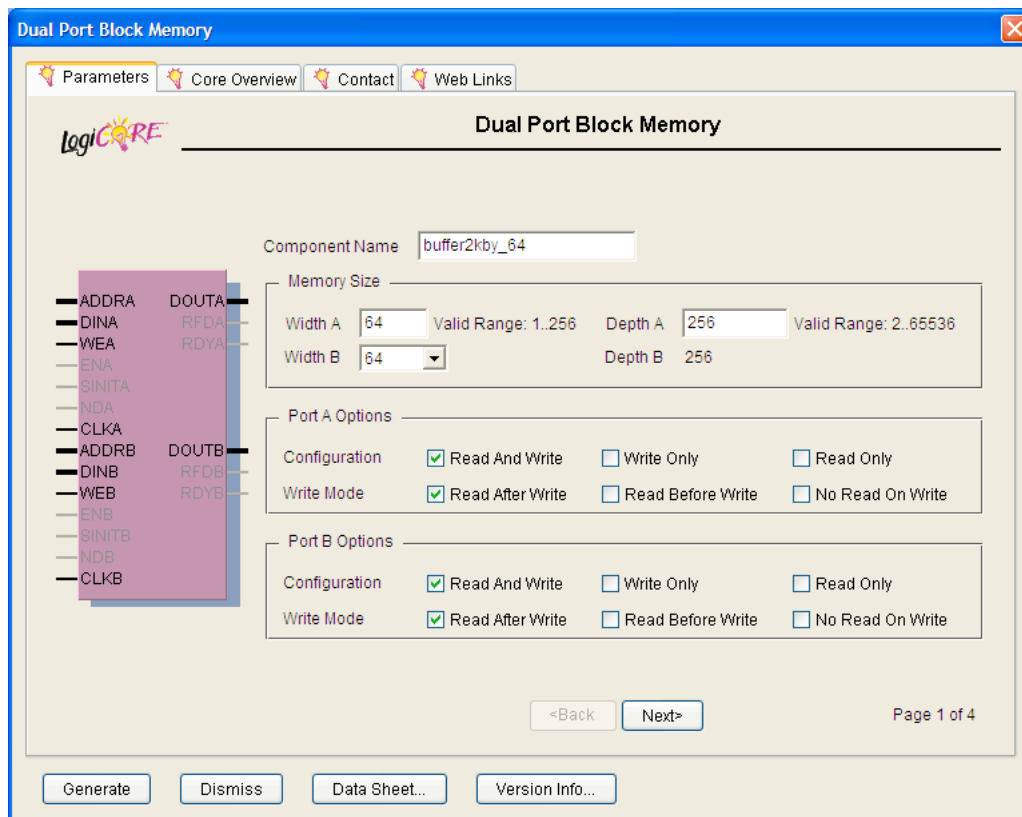


Figure 16: Xilinx CORE Generator Wizard

These buffers require a library called XilinxCoreLib which has to be compiled with the Xilinx Simulation Library Compilation Wizard. This tool compiles the required libraries for a given architecture, in our case the Xilinx Virtex-4 FPGA.

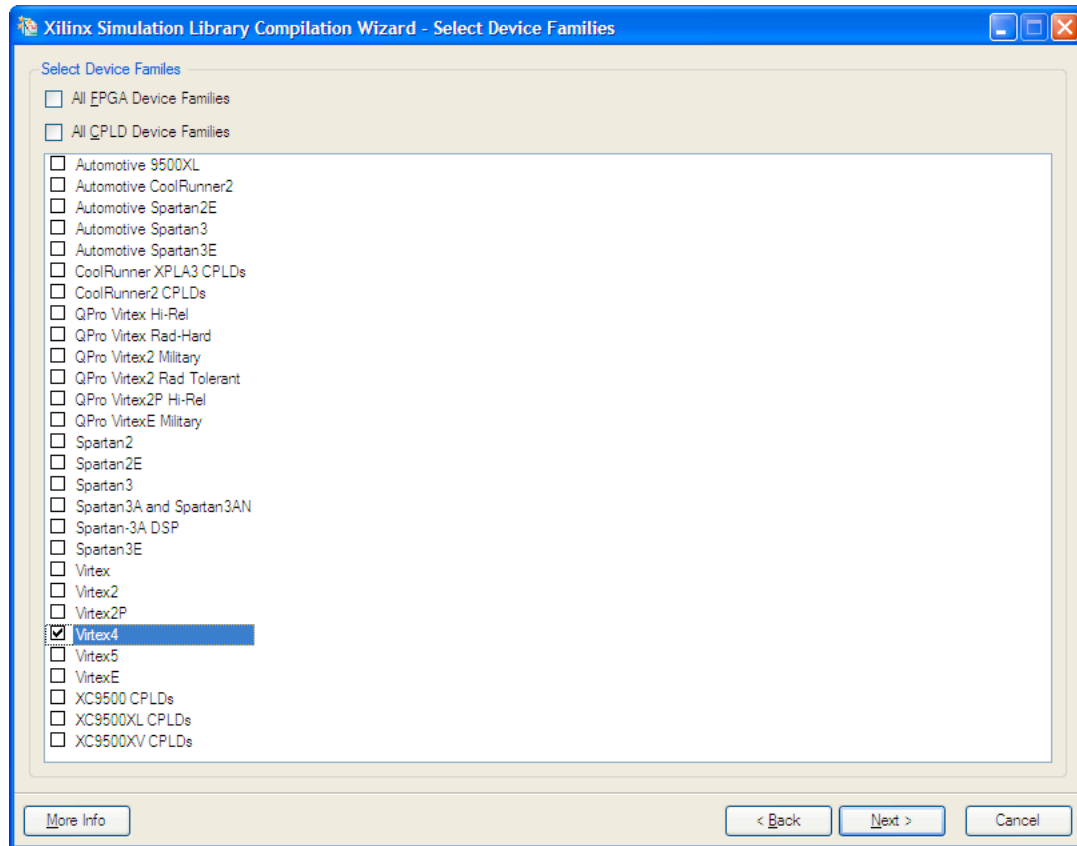


Figure 17: Xilinx Simulation Library Compilation Wizard

5.3 The top level

The main component of this HDL Designer project is called `topLevel` and it is in the `mp_lib` library. All other components are part of the `topLevel`. The basic design of the MP has been adopted from a former project of Silvan Zahno. The following figure illustrates the design of the `topLevel`.

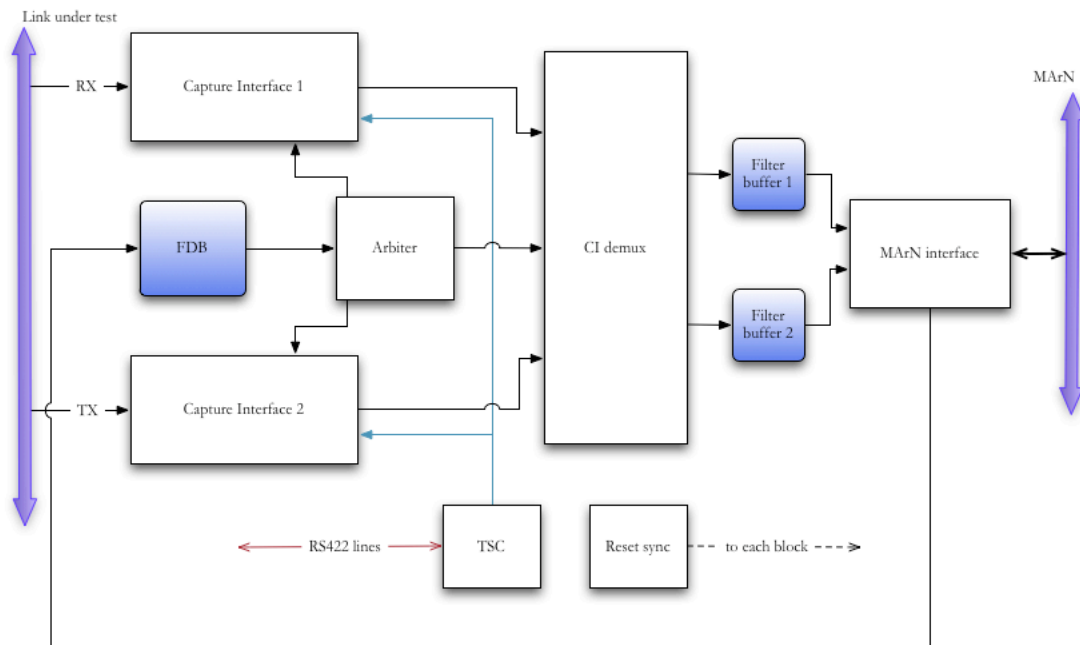


Figure 18: Schematic of the `topLevel` of the FPGA design

The following chapters describe each block that is on the schematic above. In the HDL Designer Project the simple blocks like the Reset sync and the Arbiter are in the `mp_lib` library. All other blocks are in the library called components.

5.4 Capture interface

As described in chapter 3.2 the Interface board has got two Ethernet interfaces. Those correspond each one to a capture interface (CI) of the MP. One for each direction RX and TX. Both CIs are controlled by an arbiter (see chapter 5.6). As you can see in the following figure, the CI contains mainly of three other blocks. The synchronizer, the receiver and the filter.

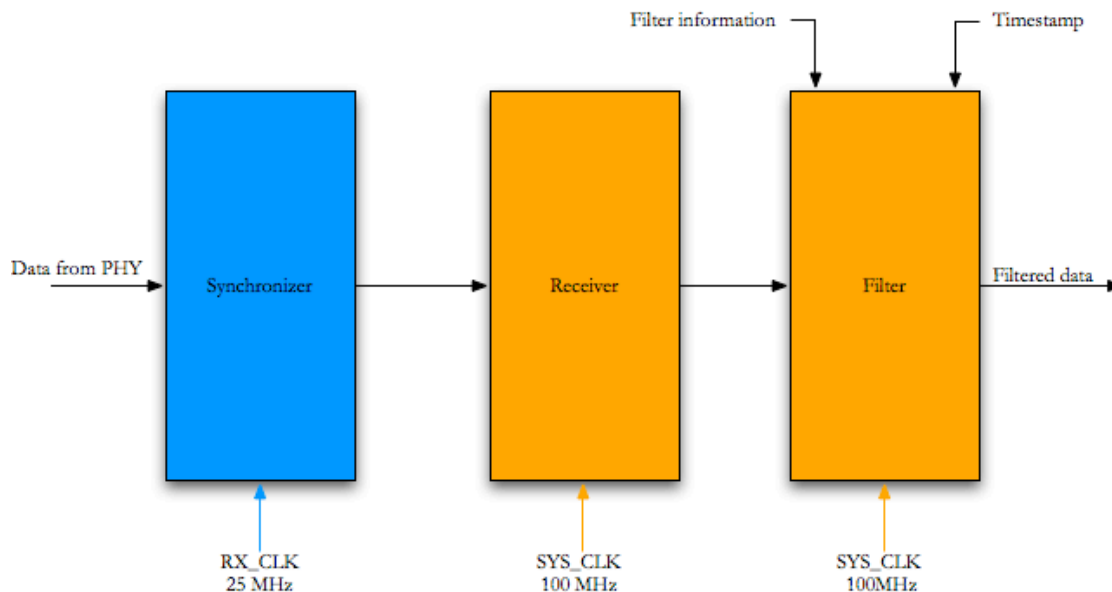


Figure 19: Schematic of the CI

As its name already gives away the synchronizer synchronises the data arriving on the PHY (the physical Ethernet interface) with its clock frequency of 25 MHz. This synchronised data is transmitted to the Ethernet receiver. From this point on the circuit is working with a clock frequency of 100 MHz. This is the default frequency of the ML405 board which is sufficient for this application. Once the Ethernet packages are in the buffer of the receiver the filter is going to read them. It filters the packages and adds the timestamp on the package that will be sent further. The filter information is coming from the arbiter (see chapter 5.6) and the timestamp from the TSC (see chapter 5.9). The only difference between the previously used Actel FPGA board and the ML405 is that the system clock has the frequency of 100 MHz instead the 50 MHz used before. But this is no problem at all. As long as the system clock frequency is higher then the RX clock of the PHY (25 MHz) the system works fine. For further explanations on the CI please see the chapter 6.2 of Silvan Zahno's diploma work report (appendix 1).

5.5 Filter database (FDB)

The filter database is a buffer that stores the filter rules. A filter has a mask and a value. Both fields are 16 bit wide. Because this MP uses two filters the buffer has a 64 bit data-bus. As the filter is 84 bytes long both filters use 21 RAM lines. This buffer was realised with a dual port RAM generated with Xilinx CORE Generator as described in the remark of chapter 5.2.

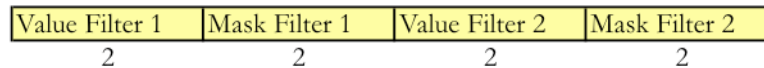


Figure 20: RAM structure FDB

Each of these four fields is 2 bytes long. The value field contains the desired package content and the mask field indicates which bits should be checked.

5.6 Arbiter

Basically the arbiter's job is to decide which one of the two CIs has the permission to be active. There is a round-robin mechanism implemented to do this decision. In this block no modifications have been made. For further information about it please see the chapter 6.2.6 of Silvan Zahno's diploma work report (appendix 1).

5.7 CI Demux

The CI demux is responsible to add the measurement header MH to a measurement frame and to start a new frame if the maximal length is reached. It takes the captured frames from an active CI including the capture header CH and stores them into a buffer. If the buffer is full, it adds the MH and informs the MArN interface to send the frame. Also in this block no changes have been made. See the chapter 6.2.7 of Silvan Zahno's diploma work report (appendix 1) for more information about this block.

5.8 MArN interface

5.8.1 Introduction

Because the MP should be controlled by the Measurement Area Controller (MArC) through the MArN it is necessary that there is an Ethernet interface on the MP which is capable to send and receive data. The transmitter is already implemented in the previous design. It was located in a block called "sender". This design is partially reused and the Ethernet receiver is implemented at the same location. This new block is called "MArN interface". The MArC will also send new filters to the MP so there are many other changes required in the design. The Ethernet interface (receiver and the transmitter) are made by the HES-SO in Switzerland and are adopted into this project. To have more information about this block, please see the appendix 8. See the following figure to get an overview of the new design.

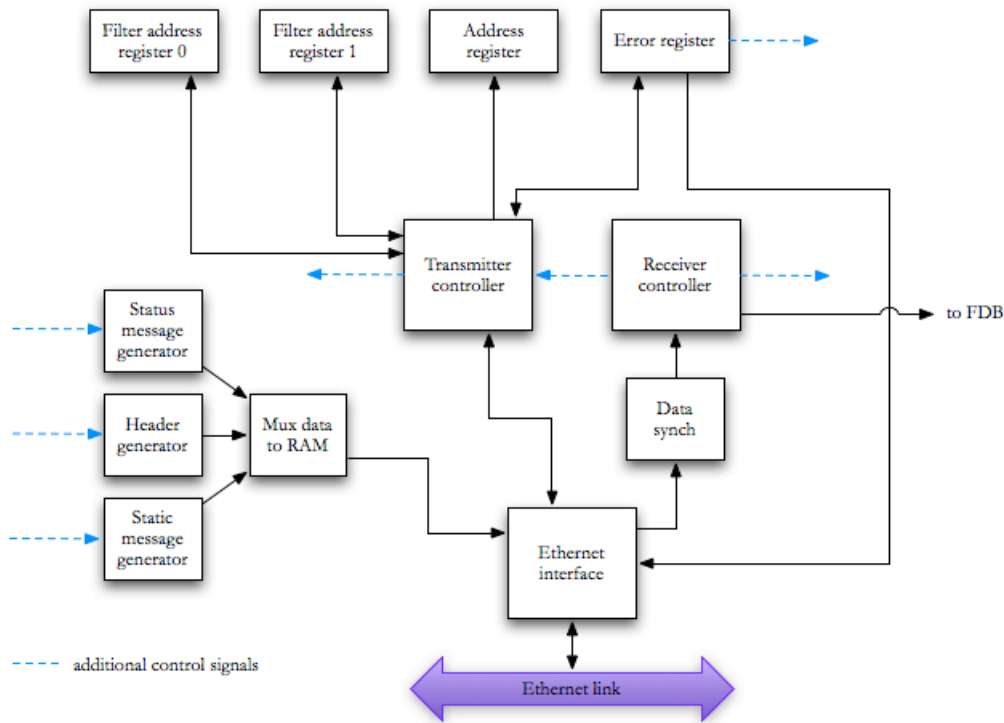


Figure 21: Schematic of the MArN interface

5.8.2 Functionality

The heart of the MArN interface are the two statemachines in the middle of the schematic above. The transmitter controller (transmitter_ctrl) and the receiver controller (receiver_ctrl). The concept of the transmitter_ctrl is mainly adopted from the previous design. There just have been made some modifications that make it possible for the MP to send not only measurement frames and status messages but also other frames like DHCP or ARP. The receiver_ctrl checks every packet that is arriving on the PHY and takes some consequences if necessary. For more information about this block, please read the chapter 5.8.4. The data synchronizer simply synchronizes the incoming data to the system clock. The receiver_ctrl manages two important local signals. The transmit_mode signal, which contains information about what data has to be sent and the static_msg_id, which indicates to the static message generator what kind of message it has to generate. The following tables show the codes that have been defined for these signals.

Value	Description
1	Send a static message. (DHCP, ARP)
10	Normal operation mode. Sending measurement frames and status messages.

Table 11: Codes for the transmit_mode signal

Value	Description
1	DHCPDISCOVER (see chapter 4.5.2.1)
2	DHCPREQUEST (see chapter 4.5.2.3)
3	ARP announcement (see chapter 4.6.2)

Table 12: Codes for the static_msg_id signal

The normal operating mode corresponds to the functionality of the previous design. That means measurement frames are sent and once every second a status message is generated. If there is a static message to send the static message generator checks the value of the static_msg_id signal and generates the corresponding message.

5.8.3 Start-up procedure

With all the protocols described in chapter 4 the MP can initialize itself and get an authorization of the MArC. All this is done in the MArC interface and will be recapitulated in this chapter. The following figure shows the start-up procedure.

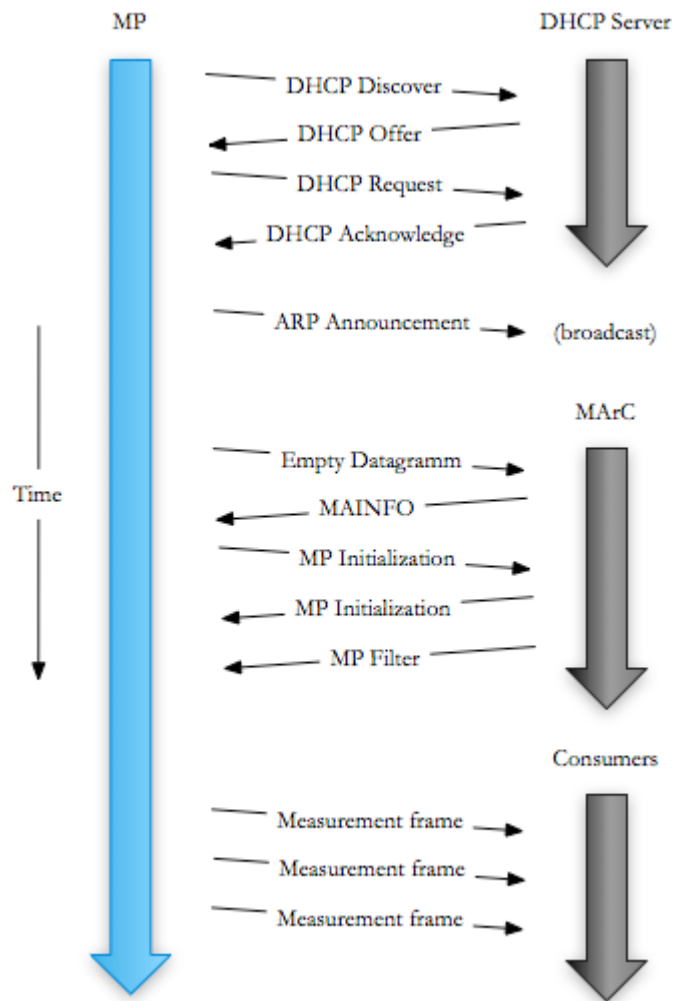


Figure 22: Start-up procedure

First the MP gets an IP address from a DHCP server. Then it makes an ARP announcement to communicate its IP address to the other devices in the MArN. As soon as that is done the MP looks for a MArC. Therefore it simply has to send an empty UDP package and the MArC will respond. After the MP has all its filters and is authorised to capture data it will start to send measurement frames to the MArN as multicast packages and the consumers will decide if they want to use this data or not.

5.8.4 Receiver controller

As the start-up procedure is now known, it is possible to design the statemachine of the Receiver controller. The following graph of the statemachine of the receiver_ctrl does not show every single state that actually exists in the statemachine. It just shows the most important states to understand the functionality of this block.

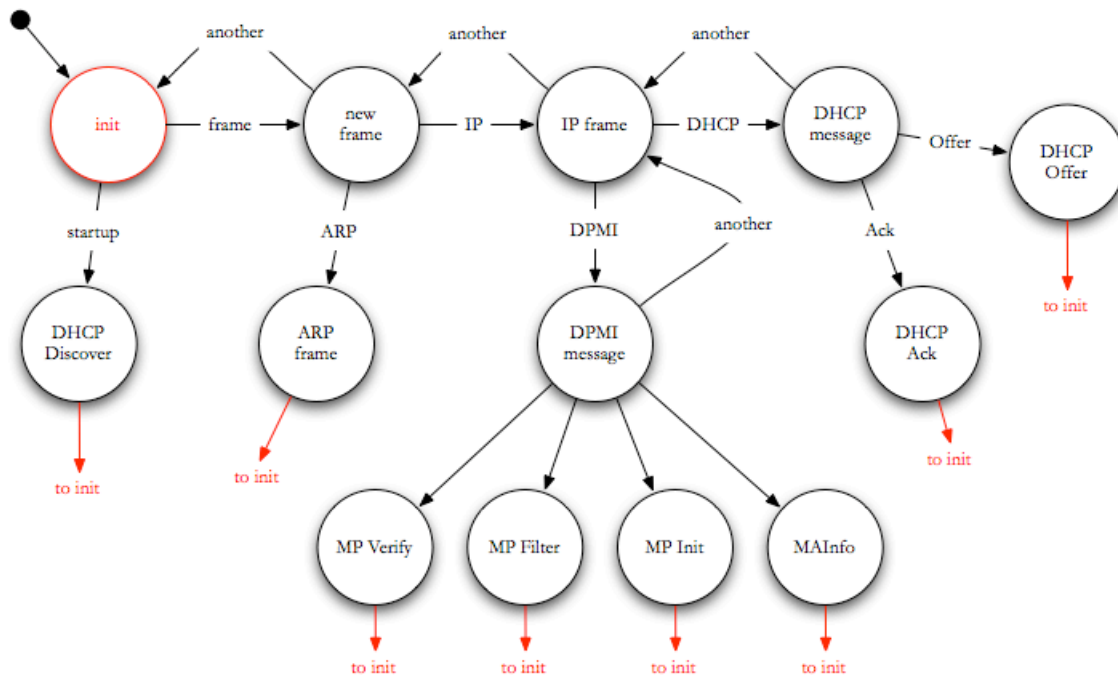


Figure 23: Graph of the statemachine of the receiver_ctrl block

On the start-up of the MP the signal dhcp_discover is set to 0. If this is the case the DHCPDISCOVER message will be generated and this signal will be set to 1. From this moment on every received package will be analyzed in the state new_frame. If the Ether-Type of the arrived package corresponds to ARP or IP the statemachine will go to the corresponding state. In the state IP_frame it will be checked if it is a DHCP or a DPMI message. This is possible while analyzing the UDP port number. The DHCP messages can be identified with their options in the UDP data (see chapter 4.5) and the DPMI messages have all different IDs (see chapter 4.2.3). In every one of the final states in the diagram above the necessary steps to generate an appropriate response message or to take an action will be initiated. Then it goes back to the init state where the next package will be received.

5.8.5 State of development

The previous sender block has been improved and is now called MARn interface. The design has been created as presented in the introduction of this chapter. The adaptations in the transmitter_ctrl have been made, so that it is possible to switch between the different transmit_modes and to send static messages. The DHCP- and the FDB update-functionality could not be implemented due to a lack of time. According the DHCP frames it is to consider that they contain manufacturer-specific options that first have to be clarified before it is possible to develop the different message decoders and generators. For the FDB update functionality a data line to the converter_filter block must be implemented and the block must be modified so that it is possible to change the data in the FDB buffer.

5.9 Time synchronization client (TSC)

5.9.1 Introduction

The first goal was to implement a time synchronization client (TSC). That means that the signals generated by the TDS (see chapter 4.3) have to be interpreted correctly and a timestamp has to be generated out of it. The timestamp has to look like it is described in chapter 4.2.2.5. The basic concept of the TSC was already developed by Silvan Zahno. It consists of three blocks. clock_drift_correction, delay_calc and a pico_counter. I added an additional block called time_decoder to decode the time signal from the TDS.

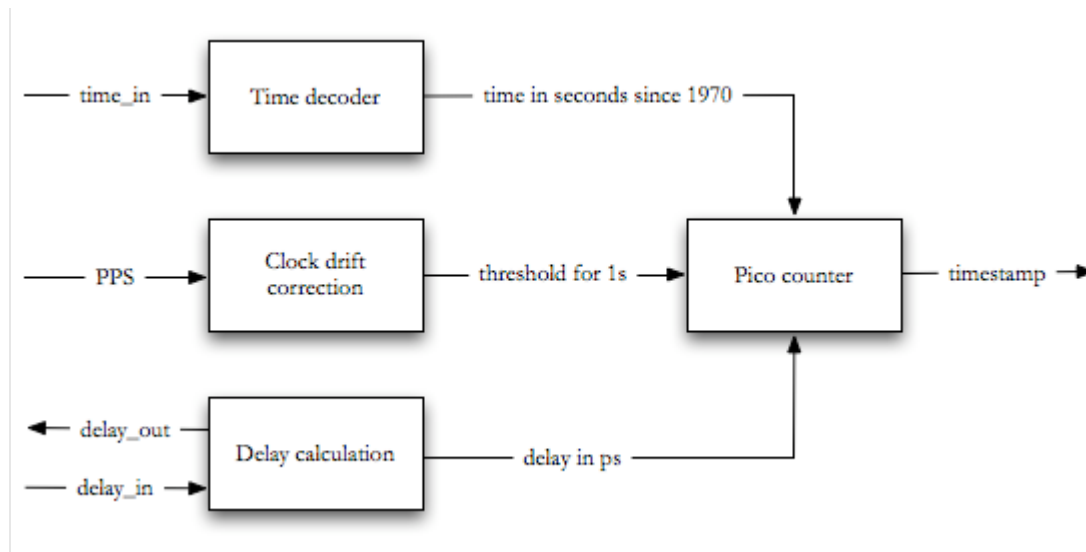


Figure 24: Schematic of the TSC client

5.9.2 Clock drift correction

The clock_drift_correction block is responsible to supply the pico_counter with a threshold-signal. A counter starts on a rising edge of the PPS-signal (pulse per second) that comes from the TDS and counts until the next rising edge. So the number of clock impulses during one second is known. This value is laid on the threshold output. With a clock frequency of 50MHz this value should be $50 \cdot 10^6$. That is why this is the initial value of this signal.

5.9.3 Delay calculation

The delay_calc block figures out with which delay the PPS arrives at the MP. Therefore a small statemachine is implemented. A signal is sent to the TDS and as soon as it arrives there, the TDS will send another signal back to the MP. While measuring the time in between it is possible to find out the delay of the PPS as it is transmitted on the same physical line. The following figure shows this statemachine.

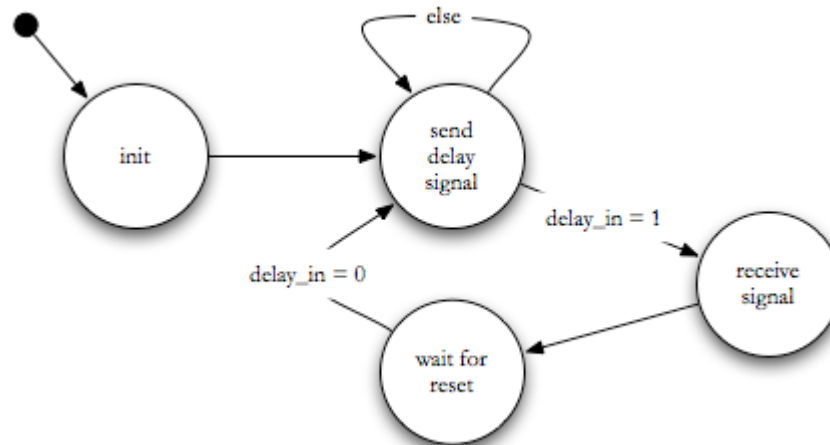


Figure 25: Graph of the statemachine of the delay_calc block

The delay value will be transmitted to the pico_counter. If it changes there is a separate signal called update_th that indicates this change.

5.9.4 Time decoder

5.9.4.1 Introduction The time_decoder block decodes the time signal which is received from the TDS and transforms it into the format that is used in the timestamp (see chapter 4.2.2.5). The TDS sends the time information using the NMEA 0183 protocol. (see chapter

4.3.3) In its definition there can be only one sender and multiple receivers. The sender transmits the data with 4800 baud on a serial line. This can be a RS-232 or like it is in our case a RS-422 serial line. For more information about the RS-422 interface please see the chapter 4.3.2. The NMEA 0183 protocol defines that the data is transmitted in the ASCII format in the form of messages. Every message starts with a dollar sign (\$) and ends with <CR><LF>. These two ASCII symbols are the carriage return and the line feed. The maximal length of such a message is 82 symbols (including the start- and end-symbols) which corresponds to 82 bytes or 656 bits. Of course there is more information transmitted that actually is used by the MP but this can be configured on the GPS device in our case the TDS.

5.9.4.2 Realization The time decoder is realized with two blocks and a D-flipflop to synchronize the incoming data stream. The following figures shows the content of the time_decoder.

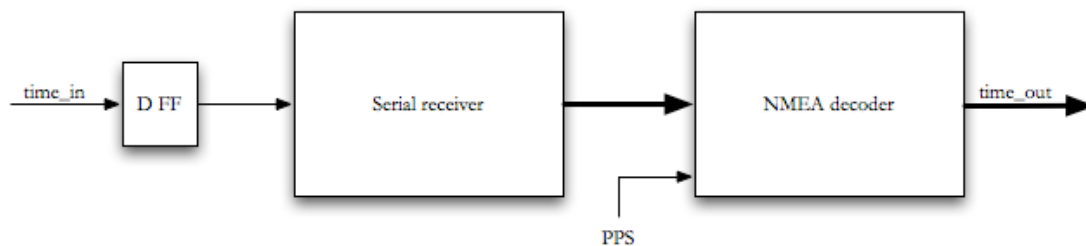


Figure 26: Schematic of the time_decoder block

After the time_in signal is synchronized it is put to a serial port receiver which decodes the asynchronous signal and provides byte-wise data on its output. Those bytes are transferred to the NMEA decoder which generates the UNIX timestamp.

5.9.4.3 Serial port receiver The serial port receiver was developed in the HES-SO in Switzerland. There is a generic called baudRateDivide which defines the baud rate of the input stream. This factor is very important for a correct signal decoding. As the NMEA signal is transmitted with 4800 baud this factor is calculated as follows.

$$baudRateDivide = \frac{systemclock}{baudrate} = \frac{100MHz}{4800baud} = \frac{100MHz}{38400 \frac{bit}{s}} = 2604.7 \approx 2605$$

With this information the serial port receiver is able to put every byte that is transmitted on the serial line on a 8 bit bus. If the value is ready on the dataOut bus it makes an impulse on the line dataValid so the next block knows when to read on the data bus.

5.9.4.4 NMEA decoder As the NMEA messages are now available in bytes there is only a block missing that decodes the time and date out of a message and generates the UNIX

timestamp. That is exactly the job of the NMEA decoder. It is implemented in a statemachine. See the following figure to have an idea how this statemachine looks like.

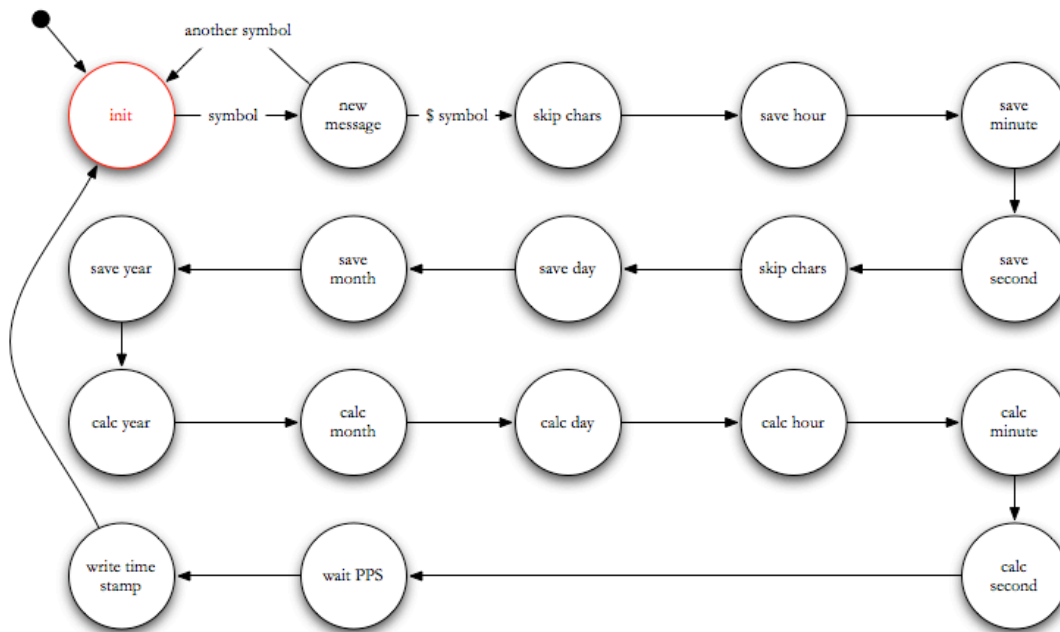


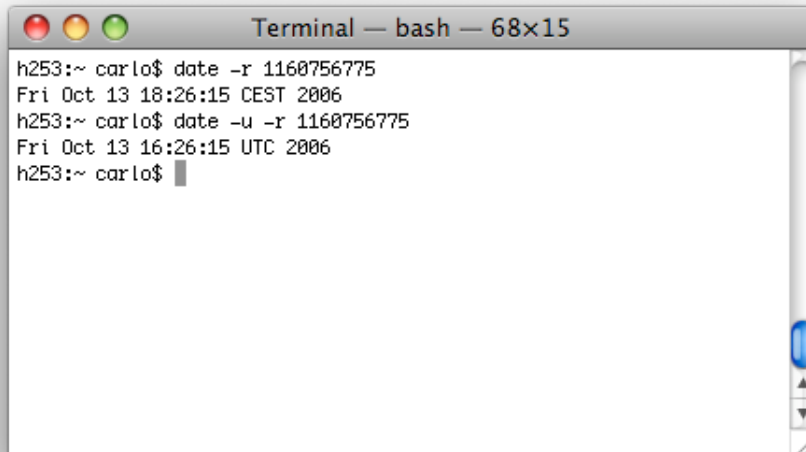
Figure 27: Principle graph of the statemachine of the NMEA_decoder block

At the beginning this statemachine waits for the \$-symbol on the data_in port. All symbols are ASCII coded. After it received this symbol it skips a certain amount of symbols until the time information is transferred. This number of skipped symbols can be configured through a generic called symbols2skip_time. For each field (the hour, the minute and the second) of the time information in the NMEA message, two symbols are transmitted. These symbols are decoded and the actual value of the time is calculated and stored to internal signals. After this there again is a certain number of symbols that must be skipped before it is possible to read the date fields. This number is again configured by a generic called sysmbols2skip_date. The date is decoded in the same manner as the time. There only is one speciality. As the year is only coded with two ASCII symbols the statemachine adds 2000 to its value. So this circuit will only work correctly until the end of the year 2999. If it would still be used by then the value in the calc_time state would have to be changed to 3000.

As the state calc_time is treated, all values of the time and the date are stored in local signals and the statemachine can calculate the UNIX timestamp. The UNIX timestamp is the number of seconds from 01.01.1970 00:00:00 until a specific date. Therefor the statemachine first counts the number of year since 1970 and adds the number of seconds

that a year contains. If it is a leap year, which is one day longer than an usual year, there will be added the corresponding number of seconds. On a similar way it is done for the months and the days. For the months it is checked how many days it has and if it is February and a leap year there must be added one day more. Finally the number of seconds passed on the current day are calculated, using the values of the hour, the minute and the second. On this stage the UNIX timestamp is calculated. To put it not too early on the bus, the statemachine waits for the next PPS. As one additional second was already added by the previous state, the correct timestamp will be on the bus. The only issue would be if this calculation of the timestamp would take longer than one second but with the baudrate of the NMEA signal and the system clock of 100MHz there is no problem at all. The maximal length of a NMEA message is 82 bytes and with 4800 baud it takes $\frac{82\text{bytes}}{4800\text{baud}} = 17.0833\text{ms}$ to transfer them. So there is more than enough time to calculate the timestamp.

5.9.4.5 Tests First of all the algorithm that calculates the UNIX timestamp as described in the previous chapter had to be tested. Therefore it was implemented as a MATLAB script to see if it is correct. This script can be found in appendix 9. To check if the timestamp is correct there are many functions for nearly every programming language that do this conversion. There also is the UNIX command “date” which does this with the option “-r”. To see the time in UTC which is the time broadcasted by the GPS the additional option “-u” has to be added.



```
Terminal — bash — 68x15
h253:~ carlo$ date -r 1160756775
Fri Oct 13 18:26:15 CEST 2006
h253:~ carlo$ date -u -r 1160756775
Fri Oct 13 16:26:15 UTC 2006
h253:~ carlo$
```

Figure 28: “date” command to convert UNIX timestamp

To test the VHDL design a testbench was written and a simulation was done using ModelSim SE. This NMEA message was send to the time decoder.

```
$GPRMC,162614,A,5230.5900,N,01322.3900,E,10.0,90.0,131006,1.2,E*13
$GPRMC,HHMMSS,A,BBBB.BBBB,b,LLLLL.LLLL,1,GG.G,RR.R,DDMMYY,M.M,m*PP
```

The HHMMSS part is the time and the DDMMYY part the date. On the following figure it is visible that the internal signals (hours, minutes, seconds, day, month and year) have correct values and the timestamp (first signal on the wave) is correct at the end of the simulation.

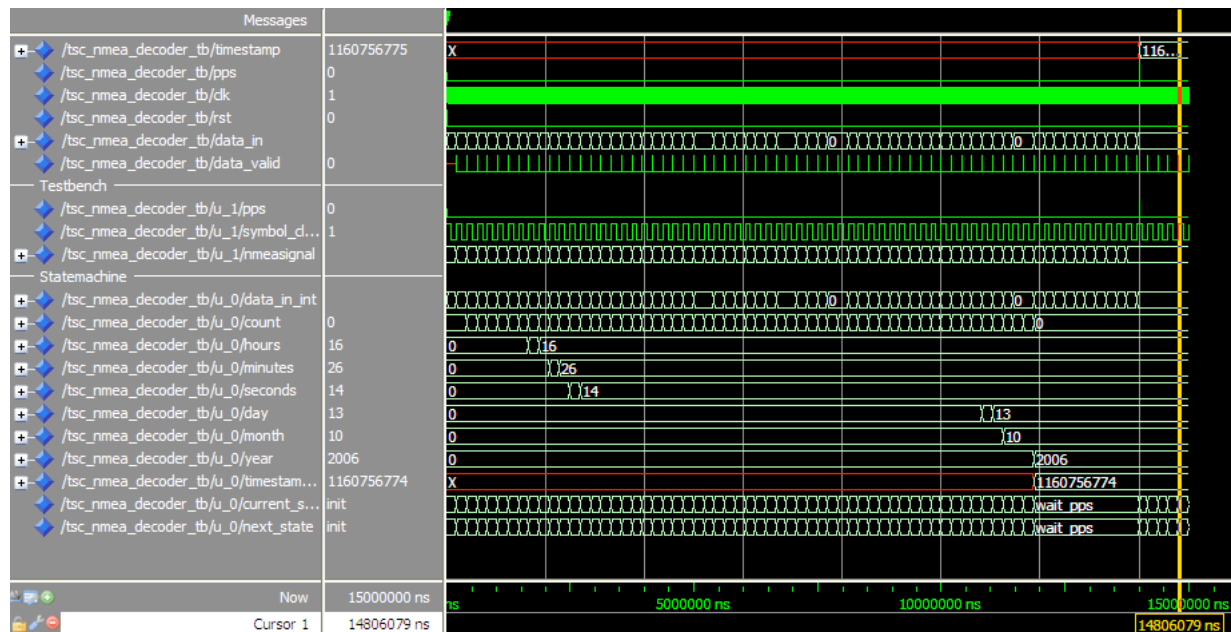


Figure 29: Wave diagram of ModelSim simulation of the time decoder

The UNIX timestamp has already added one second and is put on the bus when the PPS arrives. That is why it is 1160756775. As you can see in figure 28 this timestamp corresponds to the date and time one second earlier then the date and time in the NMEA message used in the testbench.

5.9.5 Picoseconds counter

In the pico_counter the timestamp will actually be generated. Therefor a counter increments as long as its value is less then the value of the threshold. Every time the counter increments the timestamp will increment for 10000ps. This corresponds to one period

($T_{period} = \frac{1}{f_{clk}} = \frac{1}{100MHz} = 10ns = 10000ps$). Furthermore the seconds field of the timestamp will be taken from the time input and included on the first 4 bytes (32 bits) of the timestamp.

5.10 Reset synchronization

This block is a synchronizer of the reset impulse. This means the reset button is connected only to this block. If it is pushed this signal is going to be synchronized to all clock frequencies used in this circuit. These are the system clock and both RX clocks of the CIs. For additional information please see the chapter 6.2.3 of Silvan Zahno's diploma work report (appendix 1).

6 Own TDS (GPS board)

6.1 Introduction

Basically for testing reasons it would be practical to have a simple TDS device. That is why a own TDS called the GPS board is developed. As described in chapter 4.3 it receives a GPS time signal and forwards it on a RS-422 line to the MP as it is also the case with the Endace TDS-2. As GPS receiver the Navman Jupiter 30 GPS controller has been chosen. Please see the appendix 12 for further information. This controller can be bought on a module that includes an antenna connector and a header with all needed signals. To treat the information that is supplied by the GPS a Xilinx Spartan FPGA is used. This FPGA is available on a test board of the HES-SO.

6.2 RS-422 interface

To synchronize the time on the MPs there is a RS-422 interface on each MP. As further projects at BTH Karlskrona already dealt with this problematic it is desired to have the following four differential lines available.

Signal name	Description	Direction
Delay In	A periodic pulse that the client generates to calculate its distance to the master.	In
Delay Out	A pulse arriving on the “Delay In” line will be sent back to the client on this line.	Out
PPS	Pulse per second signal that is received over the GPS.	Out
Time	The absolute time coded in a certain format.	Out

Table 13: TDS signals (masters view)

So the number of line drivers and receivers needed is given by one receiver and three drivers (transmitters). A RJ45 connector is used for this scenario. The RS-422 driver needs a 5V power supply but from the FPGA test board is only 3.3V available. So the 5V must be supplied by an external power source. For more information about the PPS and the time signals please see chapter 4.3.3.

Remark: The commercial TDS-2 of Endace does not have the functionality to calculate the time-loss in the cables. So this would be an additional feature of the GPS board as it has the two line delay in and out which makes it possible to calculate this time-loss.

6.3 Physical properties

Because the FPGA is on a separated test board of the HES-SO Sion a 26 pin connector is necessary on the GPS board. The Navman GPS module can be mounted on the “main board” with four screws and the data exchange occurs through the 20 pin header. It is possible to select by a jumper either that the PPS pulse comes directly from the Navman GPS module to the RS-422 driver or goes first through the FPGA.

For the power supply two DC10B power connectors are mounted on the board. They are connected in parallel to have the possibility to supply another device with 5V.

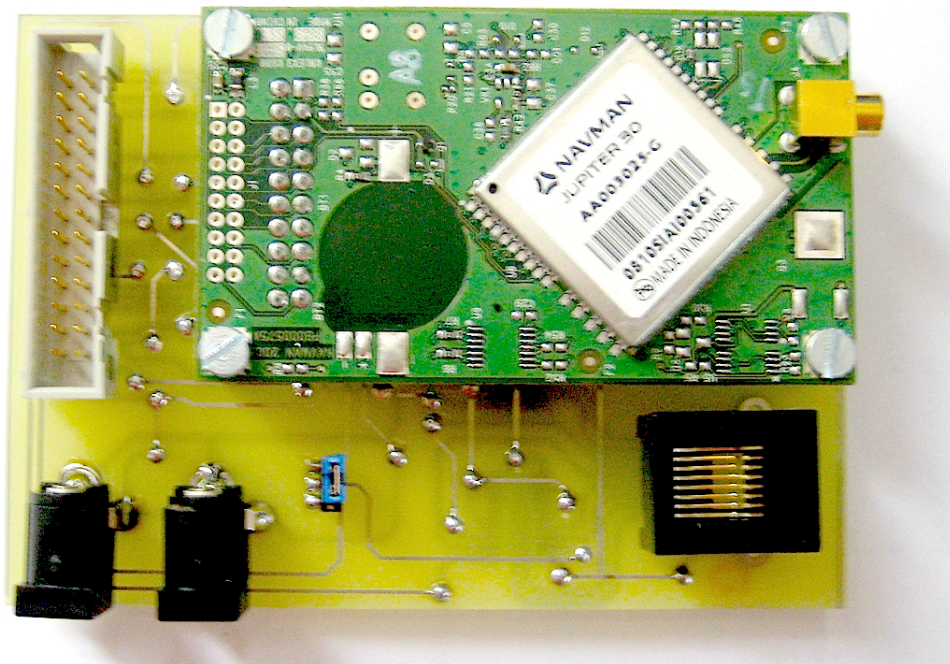


Figure 30: GPS board

6.4 Realization

The GPS board is connected to a FPGA test board of the HES-SO called FPGA-EBS through a 26 pin ribbon cable. The FPGA on that board is a Xilinx Spartan. The basic functionality of the FPGA is schematised in the following figure.

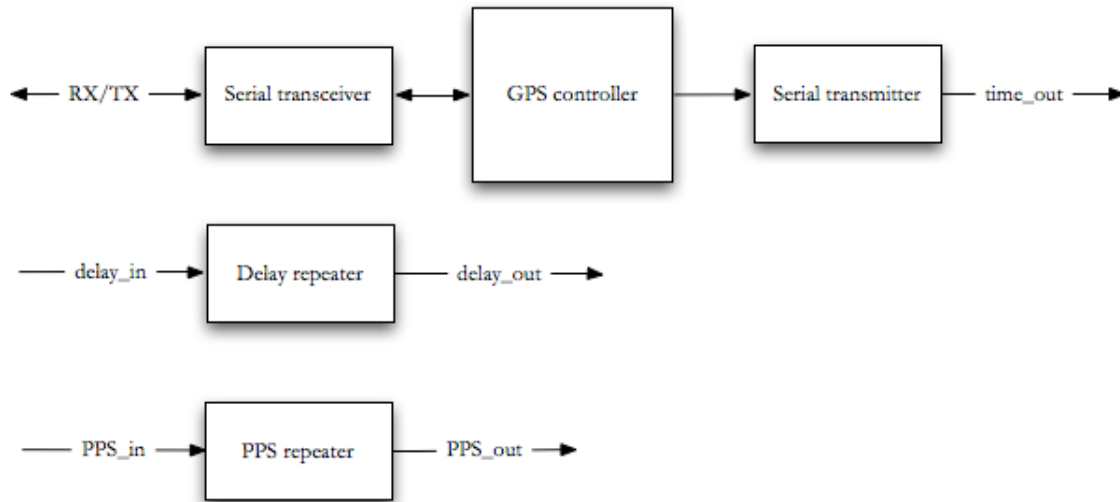


Figure 31: Schematic of the FPGA design

A serial transceiver is necessary to communicate with the NAVMAN Jupiter 30 GPS controller chip. In the block GPS controller a statemachine has to be implemented. Its purpose is to configure the Jupiter 30 chip through its serial interface in an initial phase. That means that the NMEA output of this chip will be specified. Once it is configured properly an NMEA GPS signal will be transmitted over the same line. This signal may have to be modified and then will be sent via a serial transmitter back to the GPS board where it will be sent to the RS422 interface. The delay- and the PPS-signal will be simply forwarded by the FPGA. Maybe the PPS could be used for other purposes inside the FPGA but if it is not used (like in the schematic) it is recommended to set the jumper on the GPS board in the position that the PPS is sent directly to the RS422 interface. So there will not be any delay.

6.5 Progress and stage of development

The GPS board has been developed at HES-SO and has been tested. The PPS is received and can be visualised on an oscilloscope. The following figure shows a screenshot of an Agilent 54622D oscilloscope.

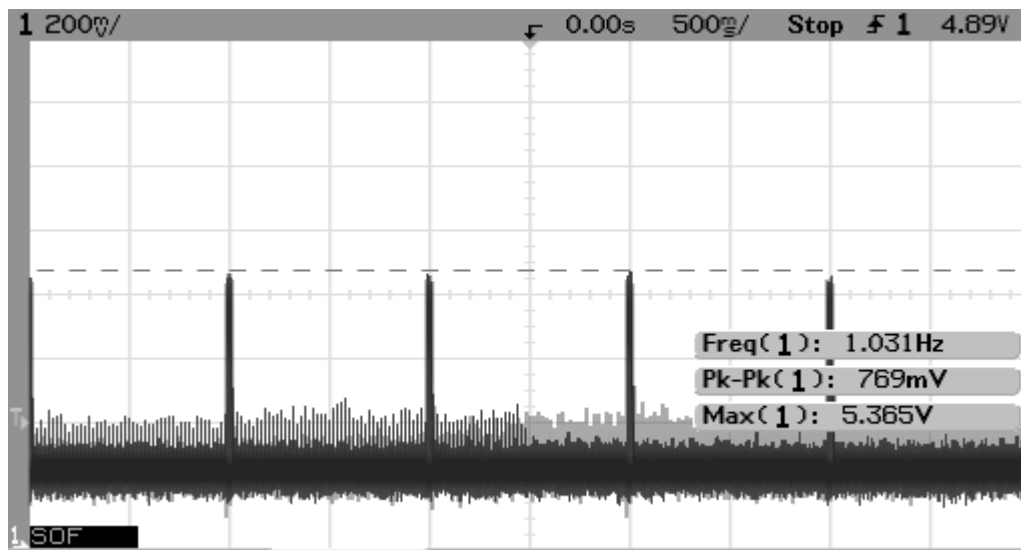


Figure 32: PPS on the oscilloscope

Because the time for this thesis was not sufficient to develop the VHDL design for the FPGA-EBS test board it was just planed theoretically as it was described in the previous chapter 6.4. The electrical schematic and the PCB layout can be found in the appendixes 10 and 11.

7 Conclusion and remarks

In this work a DPMI compatible Ethernet traffic measurement point has been developed with a Xilinx ML405 FPGA board. There have been many challenging issues during the past three months and not everything that was planned to realize could be achieved. The time for such an extensive project simply was not sufficient as there also was a lot of time needed to get knowledge about many technical particulars. I did build a Converter board to connect the Interface board to the ML405 and a TDS board (GPS board). It was possible to load the VHDL designs to the CPLD and the FPGA on the different boards. In the design of the CPLD on the Interface board it was not necessary to change anything as it is working fine. The design of the FPGA on the Controller board has been adopted and the sender block has been replaced by a block called MArN interface. The idea of this block is to communicate with the MArC. Therefor it has to be able to function as DHCP client, to make an ARP announcement and to handle ARP requests. Furthermore it has to handle DPMI messages and to send measurement frames. The following functionalities have not been developed due to the lack of time. The DHCP functionality and the ARP request handling. It is also not possible now to update the FDB. This means that the MP actually uses static filters and a static IP address. Unfortunately there was no time left to develop a design for the GPS board.

Personally this bachelor thesis has been a very good experience for me. It was very interesting and challenging to plan and develop several systems from scratch but also to adopt existent designs and improve them. Also the hardware-attached VHDL was very absorbing for me and I am glad that I could massively improve my knowledge about VHDL and programmable logic array. Sometimes I was disappointed because I was stuck on an issue for a couple of days but I learned that there always was a solution for every issue. This experience will help me a lot in my future professional life.

Because this work could be made within an Erasmus student exchange program it was a valuable experience to see another university in another country. It was very nice to meet many people from different countries and of course to get know Sweden, its people and its culture. Therefore I would like to thank to the people of the MOVE bureau in Sion and the international office in Karlskrona who made this possible. Specially I would like to thank Patrik Arlos, my supervisor and expert at BTH Karlskrona for his efforts and explications during the past three months. Furthermore I thank François Corthay, my tutor at HES-SO Sion and his assistants Silvan Zahno and Oliver Gubler for the excellent support they gave me remotely. I really appreciate it. Last but not least my thanks go to my family and my friends in Switzerland and Sweden. Thank you for motivating me all the time.

Finally, I would like to propose some further work that may can be realized by other students as projects.

- A useful feature for the MP would be if it would support more then two filters.
- Finalizing the following points of the MArN interface development:
 - Implementing a FDB update functionality. Therefor a data link between the MArN interface and the FDB buffer has to be implemented. The block that generates the static filters can be removed then.
 - Implementing a DHCP functionality. The statemachine in the receiver_controller block in the MArN interface has to be improved. The functionality is described in the chapters 4.5 and 5.8.4.
 - Implementing a functionality that handles ARP requests as described in chapter 4.6.3. (ARP announcement is already implemented)
- As already mentioned the implementation of the GPS board has to be done as proposed in chapter 6.4.

Karlskrona, 18 August 2009

Carlo Arnold

8 Appendixes

- 1 Diploma work report of Silvan Zahno
- 2 Electrical schematic of the Converter board
- 3 PCB layout of the Converter board
- 4 Xilinx ML405 user guide
- 5 DPMI documentation of BTH
- 6 Header file of a software MP in C
- 7 Endace TDS-2 documentation
- 8 IO MAC Ethernet interface documentation
- 9 MATLAB script for UNIX timestamp calculation
- 10 Electrical schematic of the GPS board
- 11 PCB layout of the GPS board
- 12 Navman Jupiter 30 data sheet

9 Glossary

ARP	Address Resolution Protocol, is a protocol to find a host's hardware address if only the address of the higher layer is known.
ASCII	American Standard Code for Information Interchange
BTH	Blekinge Institute of Technology, Sweden
CPLD	Complex Programmable Logic Device, is a programmable logic device. The building block of a CPLD is the macro cell, which contains logic implementing disjunctive normal form expressions and more specialized logic operations.
CRC	Cyclic Redundancy Check, is a non-secure hash function designed to detect accidental changes to raw computer data.
DHCP	Dynamic Host Configuration Protocol, is a network protocol that allows devices to obtain their configuration information to operate in a IP network.
DPMI	Distributed Passive Measurement Infrastructure, is a concept developed by Patrik Arlos at BTH Karlskrona.
FPGA	Field Programmable Gate Array, is a semiconductor device containing programmable logic components and programmable interconnects.
GPS	Global Positioning System, is a global navigation satellite system that also broadcasts a precise time signal.
HES-SO	University of Applied Science Western Switzerland
IP	Internet Protocol, is a network protocol used to communicate data through the Internet or local area networks.
JTAG	Joint Test Action Group, is the usual name used for the IEEE 1149.1 standard entitled Standard Test Access Port and Boundary-Scan Architecture for test access ports used for testing printed circuit boards using boundary scan.
MAR	Measurement Area, is defined in DPMI.
MARc	Measurement Area Controller, this device controls the MP and the consumers of a MAR.
MARn	Measurement Area Network, network which analyzes and tread the sent Measurement frames.
MP	Measurement Point
NMEA-0183	Protocol developed by the National Marine Electronics Association (NMEA) for data exchange of devices used by the marine.
PPS	Pulse Per Second, is a electrical signal that makes an impulse on the start of a second.

- RJ45 RJ, is a standardized physical network interface and RJ45 is the commonly used interface used for Ethernet.
- RS-422 Former name of EIA-422 which is an electrical standard to transmit data.
- TDS Time Distributing Server, is a server that supplies clients with an exact time signal usually over a RS-422 line.
- TSC Time Synchronization Client, is the part of the MP that generates the timestamp that is included in a measurement frame out of a time signal from a TDS.
- UDP User Datagram Protocol, is a protocol of the IP suite and belongs to the transport layer.
- VHDL Very High Speed Integrated Circuit Hardware Description Language, is a programming language to describe hardware.