Christian Halbfurter, BSc

High Speed Data Processing for Digital Printing Systems

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Ao.Univ.-Prof. Dipl.-Ing. Dr. techn. Eugen Brenner

Institute for Technical Informatics

Graz, May 2017

Kurzfassung

Die Österreichische Firma Durst Phototechnik Digital Technology GmbH aus Osttirol ist Marktführer für industrielle high-performance Drucklösungen. Ihr Key-Segment liegt im Bereich des Large Format Inkjet Drucks. Das Portfolio umfasst weiters Textil-, Keramik-, Glas- und Label-Druck.

Konventionelle Drucksysteme verwenden das Multipass Verfahren, wobei sich der Druckkopf mehrfach über das Druckmedium bewegt um das komplette Bild zu produzieren. Ein anderer Ansatz, das Single-Pass Verfahren, verwendet statische Druckkopfreihen, welche kontinuierlich Farbtropfen auf das sich darunter voranbewegende Druckmedium abfeuert. Somit kann die Druckgeschwindigkeit um ein Vielfaches erhöht werden.

Daraus resultieren auch höhere Datenraten und komplizierteren Elektronikschaltungen. Um die korrekte Funktion der Kontrollelektronik gewährleisten zu können, ist ein Testmodul notwendig. Aus diesem Grund behandelt diese Arbeit die Entwicklung eines solchen Testmoduls für die high-speed Datenerfassung und Verarbeitung. Mit diesem Modul kann die Kommunikation zwischen Kontrollelektronik und Druckkopf auf korrekte Übertragung und Datenintegrität überprüft werden. Um die großen Datenmengen verarbeiten zu können, wurde ein Xilinx Zynq SoC verwendet. Neben den high-speed Druckdatensignalen wird auch die Anstiegszeit und Amplitude der Tintendüsen-Zündpuls-Spannung überprüft.

Die Programmierung der Zynq Hard- und Softwarekomponenten erfolgt mithilfe der Xilinx Vivado Design Suite und SDK. Um mit einem Proof-of-Concept die Kernkomponenten verifizieren zu können, wurde in der ersten Prototypenphase ein PicoZed System-on-Module verwendet. Die darauf folgende Portierung auf das finale Testmodul PCB und die Integration in die Drucksystem-Umgebung wird als Folgearbeit ausgeführt.

Abstract

The Austrian company Durst Phototechnik Digital Technology GmbH located in Eastern Tyrol is a market leader in high performance industrial printing solutions. Their signature segment is the large format inkjet printing, while the portfolio also includes segments as textile, ceramics, glass and label printing.

Conventional printing systems use a multipass approach, where the print-head moves along the printing medium multiple times to produce the whole image. Another approach is the single-pass technology, where a static array of print-heads continuously fires the color drops while the medium passes the print-heads beneath. With this system the printing speed can be increased many times.

This also results in higher data rates and more complex electronic circuits for the print-head control system. To verify the correct functionality of the control electronics a test module is necessary. This work considers the development of a test module for high-speed data acquisition and processing. The test module is able to verify the signal integrity and correct data transmission from the control electronics to the print-heads. Due to the high data rates a Xilinx Zynq SoC handles the data processing. Beside the high-speed data signals for the print-head also the amplitude and rise time of the fire-pulse signals for the ink nozzles are verified.

The programming of hardware and software components for the Zynq SoC is done using Xilinx Vivado Design Suite and SDK. For a proof-of-concept and to verify the core functionality in the first prototype phase, the PicoZed system on module is used. The porting to the final test module PCB and printing system infrastructure will be a follow up work.

AFFIDAVIT

Acknowledgment

This master's thesis was accomplished 2016/17 at the Institute for Technical Informatics at Graz University of Technology.

First I want to thank the company Durst Phototechnik Digital Technology GmbH, especially DI Peter Weingartner for enabling me the great opportunity of this work, DI Wolfgang Knotz for his special commitment in organizational tasks and DI(FH) Mario Ploner for his professional technical support.
Furthermore I want to thank Prof. Eugen Brenner of the Institute for Technical Informatics for his excellent supervision and mentor work, which at the end helped me a lot in finishing this thesis in such a sophisticated way.

I also want to thank my girlfriend Theresa for her patience and mental support and my parents for their appreciation and overall support during my study.

Graz, in May 2017 Christian Halbfurter

Contents

List of Figures

List of Tables

Chapter 1

Introduction and scope of work

1.1 Motivation

The industrial printing market is highly influenced from digitalization. To be able to keep the position of a market leader in industrial printing solutions, the Austrian company Durst Phototechnik Digital Technology GmbH[1] drives the innovation process further. To fulfill the customer needs of high printing speeds, new printing technologies were developed.

Away from the traditional way of printing with multipass systems, where the print-head itself moves ahead the medium, the single-pass approach was invented. Instead of moving print-heads, the system consists of arrays of print-heads seamlessly attached to each other. The printing medium just slides below the static print-head arrays and thousands of nozzles fire their color drop at the right time. With this technique the printing speed can be increased multiple times, but it also increases the complexity of the control electronics for the print-heads.

To guarantee the customer high quality printing results over a long product lifetime, a test module for the control electronics is necessary. Due to the high-speed of data flow, the control electronics embeds an Xilinx Artix-7 family FPGA on the PCB. For the data control and user interface a Xilinx MicroBlaze soft-processor core is used. This well-tried HW/SW-codesign architecture currently fulfills both, the company and customer needs. For more complex data processing purposes Xilinx introduced the Zynq platform, which has a hard-IP ARM dual core processor besides the programmable logic on a single chip. With this powerful SoC the drawbacks of soft-IP processors can be eliminated while keeping the advantages of the programmable logic.

1.2 Objective

In this work the HW/SW-codesign of a high-speed data acquisition and testing solution for digital printing system electronics based on the Xilinx Zynq-7000 family platform

[1]www.durst-group.com

should be implemented and evaluated. The following sections list the main aspects of the proposed embedded system:

Processing platform: Most conventional microcontrollers are not fast enough to handle such high data rates. Therefore a suitable platform has to be found to meet all project requirements.

Hardware vs. Software: As a typical HW/SW codesign process the separation of required functions is important for the performance of the resulting system. This should be considered in a comparison of different design approaches.

System integration: For a first proof-of-concept of the core-functionality an appropriate hardware platform should be found. The implementation should be accomplished to be able to easily adapt the results to the corresponding test module PCB, which was implemented in a previous work. Furthermore the whole test module should fit into the printing system's infrastructure for industrial verification and testing applications.

The implementation is done on a Xilinx Zynq-7015 SoC. In the first stage a Avnet PicoZed Board with appropriate carrier card is used for developing the core functionality. Later on the appropriate hardware and software modules will be ported to the target electronics.

1.3 Outline

In the **1st chapter** the basic digital printing technologies will be explained. A more detailed insight will be provided to the applied technology with respect to the integrated print-head of the real printing system.

In the **2nd chapter** the design phase will be described. The evaluation and selection of the best fitting processing platform, the most important design considerations and the separation of the hardware and software modules are covered here. Also a closer look will be taken to the already developed test module PCB.

The **3rd chapter** shows the resulting implementation of the test module. Besides a short overview of the PicoZed system capabilities, the test procedure with all implemented modules will be described in detail.

Chapter 2

Digital printing basics

The evolution of printing nowadays is at a level where it is not always necessary to produce predefined printing masks for each dedicated pattern. When looking back to the analog printing technology, a lot of different steps and huge effort were necessary to print one certain revision of a book. Furthermore effort was needed to integrate modifications or corrections in the printing process. With the development of the digital printing technology a huge step into very flexible, customizable printing solutions was made, which can also handle high speeds and different materials.

One of the most common sectors of digital printing is the **inkjet technology**, which can be divided into several categories. At inkjet technology it is all about how the generation of the drop is realized. A general classification can be seen in Figure 2.1 [IMH12]. Basically we can distinguish between three main groups, Electrospray, Continuous and Drop-on-Demand, whereas we will focus on the last two.

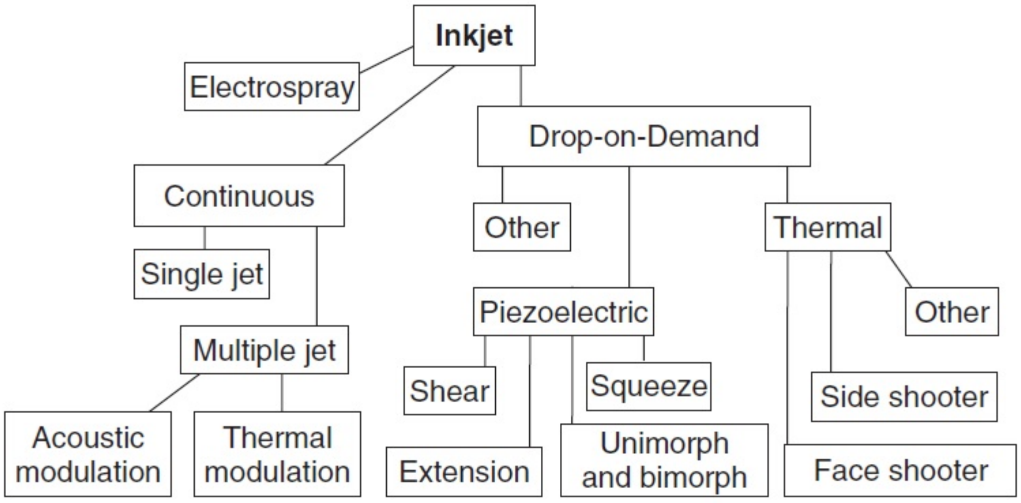


Figure 2.1: General classification of inkjet technologies

2.1 Drop generation methods

To be able to compare the different drop generation techniques it is necessary to have a closer look to the functionality and structure of them. In all different methods the liquid ink has to flow through a small hole, the so called nozzle. The way how the ink flows through characterizes the different methods.

2.1.1 Continuous Ink Jet (CIJ)

As the name implies in this method a continuous liquid flow, a so called jet, is formed by pressing the ink through a nozzle. This continuous jet ideally breaks up into several small drops of defined size by imposing disturbances to the jet. This resulting continuous series of drops has to be sorted out to required drops and not required drops for printing, which makes the complete process more complex. A further mechanism has to ensure the proper recovery of unused drops back to the ink supply system.

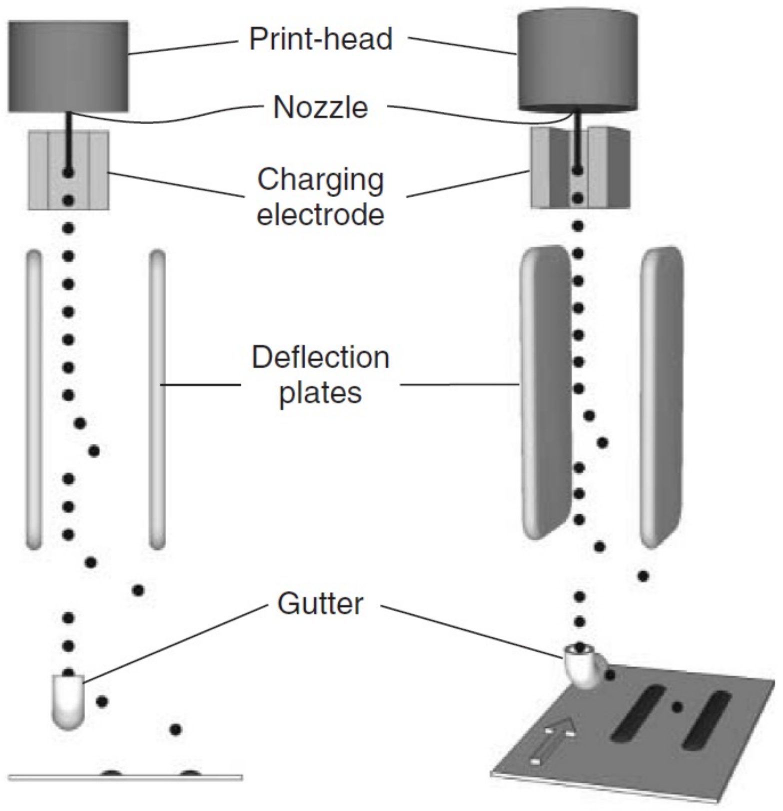


Figure 2.2: Operation principle of CIJ systems

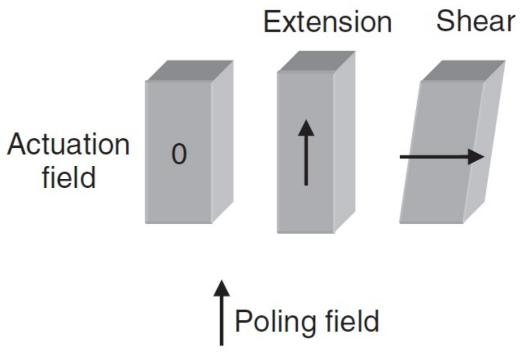
At the point where the drops are formed, mostly by vibrations from a piezoelectric structure, an electrode can charge the drops to a certain potential. When the charged drops pass through a static electric field, generated by a second pair of electrodes, each drop will be deflected by the electrical field with respect to it's potential. Uncharged drops will undergo no deflection and can directly be recovered to the ink supply system by a gutter or catcher. When moving the printing medium accordingly under the drop stream an image can be build up.

2.1.2 Drop-on-Demand (DoD)

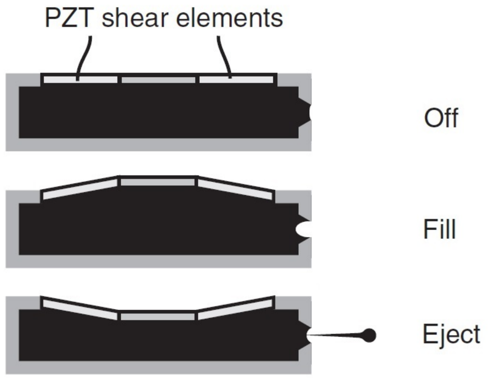
In contrast to the continuous fluid stream at CIJ, the Drop-on-Demand technique does not need drop selection or deflection systems. Each nozzle can be independently fired as it is required, which results in a very flexible print-head architecture. The DoD instead needs a certain actuation mechanism for the drop generation. Out of several developed techniques the two most commonly used ones are thermal actuation and piezoelectric actuation.

Thermal DoD actuation, also called bubble actuation, consists of a small resistive heating element behind the nozzle. With a rapid heating impulse of few micro-seconds a small amount of ink vaporises so that a miniature vapor bubble is formed. When this bubble expands enough, the ink is pressed out through the nozzle so that a drop is formed. When the bubble collapses, fresh ink is drawn into the vaporization chamber again.

At **Piezoelectric Actuation** the drop generation is caused by deformation of a piezo-electric ceramic structure. The most common material for this is lead zirconate titanate (PZT), which changes it's shape when being exposed to an electric field. The way of how the material shape deforms is dependent on the orientation of the applied electric field. Figure 2.3a shows this deformation effect of a PZT element, where the shape depends on the relation of the field to the poling direction. When the electrical field is perpendicular



(a) PZT deformation



(b) PTZ wall movement

Figure 2.3: Piezoelectric effect for drop generation

to the poling direction, the PZT element shears in a certain direction. This shearing action is used to implement movable wall elements in the ink chamber as shown in Figure 2.3b. A pair of electrodes is placed at the chamber so that by applying a certain voltage to them, the PZT wall can be used to control the ink flow through the nozzle. This principle can be scaled up to arrays of thousands of such PZT inkjet channels to form the print-head.

2.1.3 Comparison

Both technologies have advantages in their key applications. CIJ is mostly used for industrial product marking and coding. Because it was one of the first inkjet technologies invented, CIJ is a **very mature** printing technology. Due to the continuous ink flow one of the main advantages is that the **nozzles cannot dry out**. Furthermore very **high ink droplets velocity** (up to 20 m/s), high drop generation rates and printing speeds can be achieved. The **complexity** of the method due to several drop generation phases and mechanisms implies a higher effort to achieve high quality results. Especially when looking at the unused drop recycling system **special solvent regulation mechanisms** have to be applied to control the appropriate viscosity level.

At DoD the advantage of **flexible nozzle control** results into the drawback of unused nozzles. This can influence the performance over time caused by **nozzles partially drying out** up or even fail to fire. But the industry developed several strategies to avoid this such as firing all nozzles in certain period where the wasted ink of this cleaning procedure is gathered in waste collectors. This necessary nozzle cleaning leads to **higher ink consumption**. DoD is also capable of **high printing speeds** with **high quality results**, which makes it well suitable for industrial applications as corrugated, textile and large format printing.

2.2 Digital printing procedures

In the digital printing sector generally two different procedures are used, Multi- and Singlepass systems. The main difference is how the print-head is aligned to the print medium. The following two sections describe both approaches in detail.

2.2.1 Multipass systems

When one thinks of how conventional home office inkjet printers work, normally it is known that the print-head is moving together with the ink cartridges along the paper. This is basically the rough explanation of a multipass system. When looking at Figure 2.4 the process can be described in more detail. Mechanical constraints of the nozzle's physical structure limit the resolution of the print-head, so that for high resolution images multiple passes are necessary. The subfigure (a) shows the first pass of the print-head over the medium. At every further pass more free spaces get filled as shown in (b) and (c) resulting in the final image in (d). This process repeats through the entire image. For color prints the appropriate nozzles at the corresponding pixel location will be fired.

The main advantage of multipass printers are the lower costs due to less components and complexity. On the other hand this results in slower printing speeds.

2.2.2 Singlepass systems

The printing speed is probably the major strengh of singlepass printers. With a special nozzle arrangement on the print-heads also high resolutions can be achieved while passing the medium only one time during the whole process. The print-heads are statically placed in arrays, each for one CMYK color channel. To cover the whole medium width, the

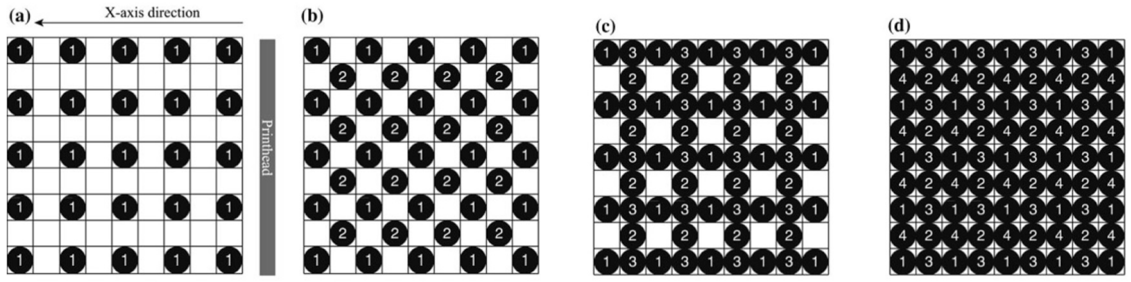


Figure 2.4: Multipass printing principle [JCW16]

print-head array must be at least as broad as the medium. Therefore also the costs of such a system increase with the postulated printing width. Especially for large format printing this technology would not be scalable. A further issue is the more complex image

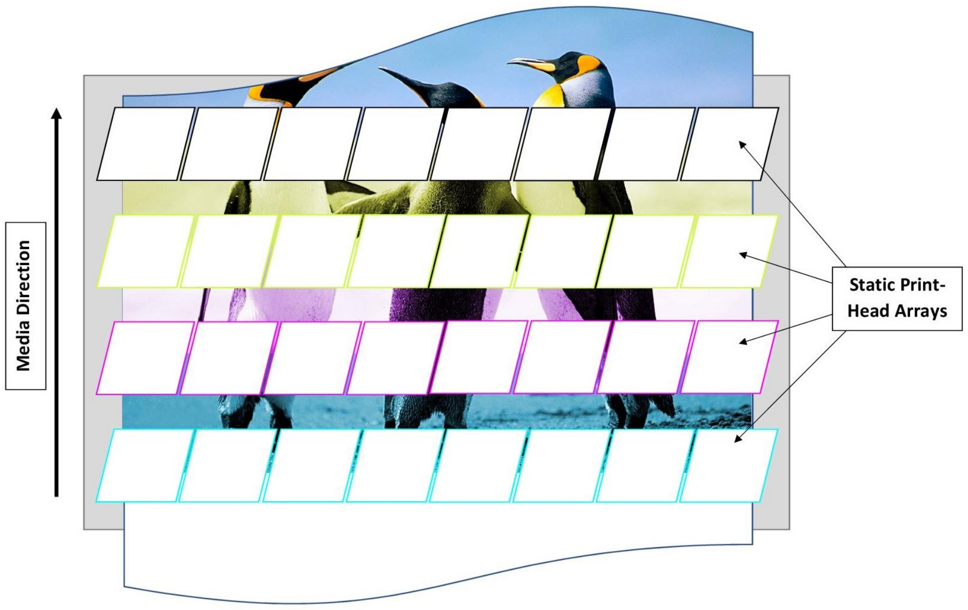


Figure 2.5: Singlepass printing process

preprocessing for the print-head control electronics. Due to the high amount of print-heads, for each pixel of the image first the according print-head position and nozzle has to be defined. Then the pixel has to be split into the four color channels. Furthermore the precise point of time for firing each color channels pixels has to be found. One can assume that this requires high-speed data acquisition and signal processing of the control electronics. The print-head manufacturers don't provide the control electronics because of mostly individual customer application needs. This can lead to development and production faults. Therefore this work is used to implement a test module for the customer control electronic's complete functionality. How this is solved for the real printing system will be covered in chapter 4.

Chapter 3

Design

3.1 Specifications

The electronic development department of the partner company defined following specifications for the test procedure:

- The most important signals to be tested are the 2x16 data lines per print-head. These signals are connected via two flex-cables to the control PCB. The high data rates of the printing system can cause unwanted crosstalk behavior. Therefore each particular **data line** has to be tested for correct data transmission.
- Beside the data lines also the signals **CLK, Latch** and **AllOn** for controlling the print head ASICs should be verified.
- The print-head has FET-drivers for firing the ink drops out of the thousands of nozzles. This fire-pulse voltage is generated by the driver stage of the control PCB. The **amplitude and rise time** of this **PZTDrive** signal should be verified. The PZTDrive rise time is between **0,5** and **2** μ**s**, the amplitude can go up to **30 V**.

The following sections describe the design procedure of the embedded test-system for high-speed data acquisition. The basic design approach was done by analyzing the test targets HW/SW requirements with respect to the design specifications.

3.2 Samba print-head architecture

Samba is an extensible *print-head on a chip* technology that is analogous to the evolution of the Integrated Circuit from a single chip with limited functionality to Large Scale Integration (LSI) incorporating thousands of integrated functions [Chr11]. With this unique print-head design a very high package density of nozzles per head can be achieved, which allows resolutions of up to 1200 dpi. The print-head is designed for high-speed single pass printing with piezoelectric DoD technology.

3.2.1 Data processing

The print-head is generally divided into two sections, front and back. Each section is electrically independent and connected by a flex-cable interface with 16 data lines and several control signals. Over these data lines the preprocessed image data is serially transfered to four 256-channel ASICs which converts the 16 bit serial bit stream into 256 bit parallel nozzle control signals. Internally each ASIC consists of four 64 bit shift registers followed by four 64 bit latches and arrays of 64 switch transistors.

In a normal print cycle for one image line first all 1024 bits for each sector have to be loaded completely (within 64 cycles) into the internal shift registers. Then the Latch signal loads the whole 2048 bits into the latch-stage to achieve a high data throughput. All logic 1 bits indicate the following switch transistor stage to fire the appropriate nozzles. With the PZTDrive signal finally the corresponding NDMOS transistors fire the jetting pulses.

$$4 \cdot 64\,\text{bit} \cdot 4\,\text{ASICs} \cdot 2\,\text{sections} = 2048\,\text{nozzles} \tag{3.1}$$

The resulting 2048 nozzles per print-head are fired up to 10^5 times per second. As mentioned in the absolute maximum ratings in [FUJ16] the input shift registers are capable to handle the 16 bit input data with a maximum frequency of 41,6 MHz. This means every 24 ns two bytes of data are shifted into the registers. To fill all 1024 bits of one print-head section 64 cycles are needed, which takes approximately 1,5μs which is equivalent to 650 kHz.

$$24.04\,\text{ns} \cdot 64\,\text{cycles} = 1,539\,\mu\text{s} \rightarrow 650\,\text{kHz} \tag{3.2}$$

$$\frac{1024\,\text{bits}}{1,539\,\mu\text{s}} = 634,8\,\text{bits}/\mu\text{s} \rightarrow 79,3\,\text{MB}/\mu\text{s} \tag{3.3}$$

This results in a data rate for one firing pulse of all nozzles of nearly 160 MB/μs per print-head. As mentioned above, the print-head is capable of printing speeds up to 100 kHz, which means firing the nozzles every 10 micro-seconds. Therefore the real data rate will be lower. Assuming single-drop pulses per fire period, within every 10μs the next 1024 bits (= 64 cycles) per print-head section have to be transfered.

$$\frac{10\,\mu\text{s}}{64\,\text{cycles}} = 0,156\,\mu\text{s for 16 bit} \tag{3.4}$$

$$\frac{16\,\text{bits}}{0,156\,\mu\text{s}} \cdot 10^6 = 97,6\,\text{Mbit/s} = 12,2\,\text{MB/s} \tag{3.5}$$

Out of Equation 3.4 one can see that every 156 ns the input shift register will get the next 16 bit of data. Following Equation 3.5 this resuls in an effective data rate of approximately 24 MB/s per print-head. When scaling up this to an array of e.g. 30 print-heads, data rates of around 730 MB/s are necessary!

To be able to transfer such high-speed data along several PCBs without losing signal quality and integrity, fiber optical interfaces and connections are commonly used. With transfer speeds up to 6,6 Gbit/s only state of the art FPGA-transceivers are capable of handling this speeds. More advanced platforms as the Xilinx UltraScale+ GTY transceiver [Xil16a] can archive speeds up to nearly 33 Gbit/s.

3.2.2 Electrical properties

As seen in subsection 3.2.1 the Samba print-head can handle very high-speed data signals. If all this signals would refer to the GND potential of the control electronics, it would make the EMC design for the control PCB very complex. Therefore the print-head has a special floating voltage level architecture. The analog PZTCommon signal is the reference potential for the nozzle fire-pulse signal PZTDrive, whereas PZTDrive is the reference for all high-speed data lines and the print-heads' 3,3 V power supply voltage. In the real

Signal	*Type*	*Reference potential*	*Purpose*
PZTDrive	Analog	PZTCommon	Nozzle FET driving pulse voltage
Data_in 0-16	Digital	PZTDrive	Nozzle on/off data
Clock	Digital	PZTDrive	Signal clock
Latch	Digital	PZTDrive	Transfer signal for buffer latch
AllOn	Digital	PZTDrive	All-nozzles-on signal
3V3 float	Analog	PZTDrive	Print-head supply

Table 3.1: Samba print-head electrical signals

printing system all these signals are electrically isolated to the control electronics. As the print-head manufacturer suggests this can be achieved by using high-speed digital isolation ICs [Lab15].

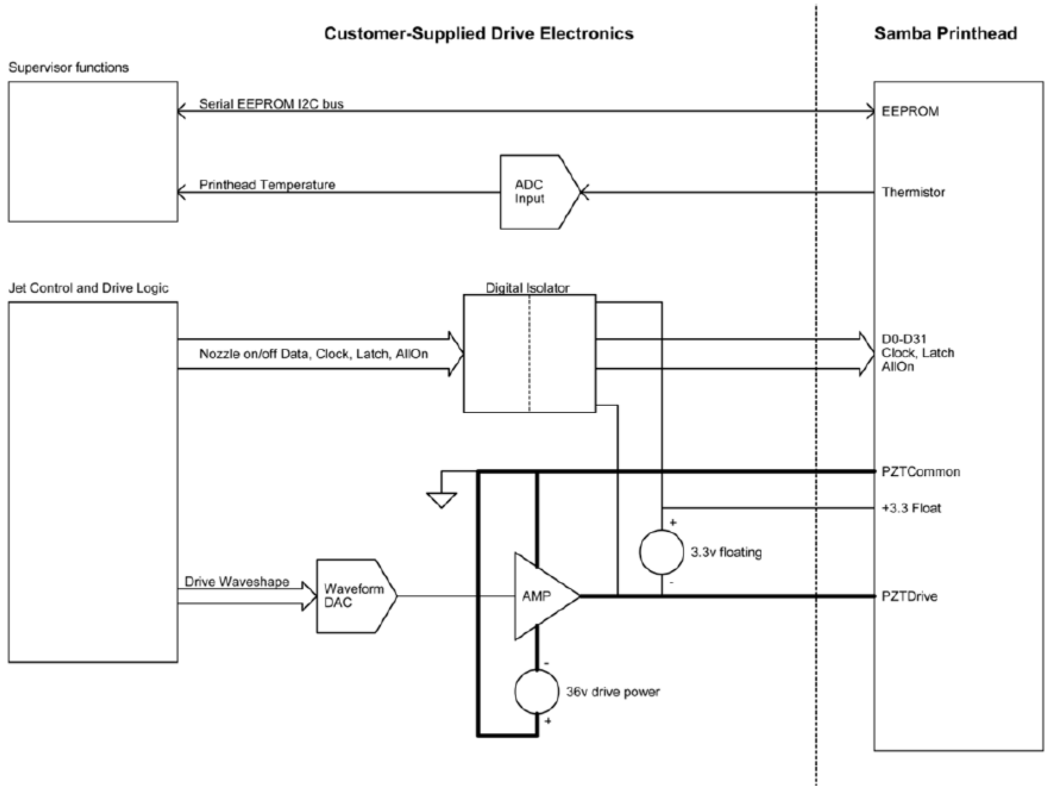


Figure 3.1: Samba print-head voltage level structure

In Figure 3.1 the Samba print-head voltage level architecture can be seen in more detail. The test module PCB proposed in [Hal17] considers all the special requirements in the hardware design.

3.2.3 Control electronics

The print-head manufacturer does not provide a dedicated control electronic PCB. Due to the different possible printing system requirements the customer has to develop its own control logic. In this case one control PCB drives four print-heads. To achieve an appropriate printing width, 30 print-heads with according control PCBs are aligned in multiple color-arrays.

Each print-head has two flex-print circuit connections for data transmission and power supply. The amount of signals can be seen in detail in Table 3.2. Before integrating the externally produced PCBs, the quality control department verifies the correct functionality of the electronics.

3.3 FPGA evaluation

Up to now the electronic development department of Durst Phototechnik only used Xilinx Artix-7 series FPGAs for their products, especially for the print-head control electronics. If necessary a MicroBlaze soft-processor core was used to process some low level control and communication tasks. The main data acquisition and processing was done in the programmable logic (PL) of the FPGA.

The first approach for the test module is to reuse the existing, well proven platform to keep the risks low and speed up the development process. The second approach with a more advanced system-on-chip as the Xilinx Zync-7000 family could be too overpowered for current applications, but furthermore could be important and necessary for future electronic development requirements. A second target of using the Zynq platform is to gain experience for alternative processing architectures.

3.3.1 Hardware requirements

For the evaluation of the proposed platforms a few hardware requirements have to be met in order to fulfill the specifications of the test target. The complete test procedure, which is described in detail in section 4.2, defines certain signals to be tested. Table 3.2 shows the amount of signals which have to be processed by the according platform.

Signal	*# per print-head*	*# per module*
Data_in	32	128
AllOn	2	8
Latch	2	8
Clock	2	8
PZTDrive	2	8
Rise time	4	16
Total	**44**	**176**

Table 3.2: Control electronic HW requirements

The resulting **176 signals** which have te be tested must be directly connected to the according input pins of the processing platform. Therefore it must have at least this amount of available input pins.

The following two sections describe different approaches for the processing platform, that can be able to fulfill this input pin constraint. Further important aspects as processing speed and on-board peripherals will be covered as well, which have a major influence in the HW/SW separation in subsection 3.4.4

3.3.2 Xilinx Artix-7 FPGA with MicroBlaze

As previously mentioned the Artix-7 series FPGA with a soft-processor IP-core is an already proven platform for the print-head electronics. Therefore only the most significant features will be discussed here.

Artix-7 series FPGA

This state-of-the-art programmable logic family comprises some best-in-class features compared to similar devices [Xil17b]:

- Sub-watt operation up to 200.000 logic cells
- Best-in-class performance per watt per dollar
- Up to sixteen 6,6 Gbit/s transceivers
- Single and double differential I/O standards up to 1,25 Gbit/s
- Best I/O to package size ratio in small form factor packages

Several device-package combinations are available. By experience of previous projects of the electronic development department a logic cell density between 35.000 and 100.000 cells/chip should be sufficient for this application. Another criterion are the available user I/Os. When considering the integrated amount of logic cells, all devices fulfill the necessary 176 signal line connections by starting at 250 I/Os. Since all Artix-7 family devices have integrated gigabit transceivers (GTP) the choice can be made by the two previous parameters. Table 3.3 shows the main parameter comparison:

Device	*Logic Cells*	*GTP modules*	*Max. User I/Os*
XC7A35T	33.280	4	250
XC7A50T	52.160	4	250
XC7A75T	75.520	8	300
XC7A100T	101.440	8	300

Table 3.3: Artix-7 device parameters

The effective parameter values depend on the chosen device package. As one can see in Table 5 in [Xil17b] the only usable package which meets all parameters is the FGG484. It features a size of 23 by 23 mm, 4 GTP channels and a maximum of 285 available user I/O pins. All devices of Table 3.3 can be ordered in this package. Hence first the smallest device **XC7A35T** could be used and simply replaced by one with more logic cells if necessary.

MicroBlaze soft-processor core

The MicroBlaze CPU is a 32-bit RISC Harvard architecture soft-processor core designed for embedded applications. It delivers more flexibility by multiple configuration options such as peripheral, memory and interface features [Kal16].

- Over 70 user configurable options
- 3- or 5-stage pipeline for flexible footprint or performance constraints
- Supports high performance AXI4 interface
- Optional Memory Management Unit (MMU)
- Optional Floating Point Unit (FPU)

The IP-core can be directly configured and integrated into the PL within the Xilinx Vivado Design Suite[2]. The most important use cases for the MicroBlaze processor are cost reduction through system integration and hardware acceleration [Kal16].

Cost Reduction Normal electronic developer rely on a broad range of electronics parts and integrated circuits. The more complex the embedded system design is, the higher the costs factor plays a role. When it comes to cost-sensitive applications a higher integration of system components can help to reduce the total costs per system. Therefore the MicroBlaze soft-processor together with the programmable logic of the FPGA core can integrate much of the system functionality in a single chip.

Hardware Acceleration Probably the most motivating reason for choosing an FPGA with an embedded processor is the flexibility of implementing tasks partially in the programmable hardware or as software function for the processor.

3.3.3 Xilinx Zynq-7000 SoC

As an alternative to a dedicated FPGA device combined with a burned-in soft-processor core, the Xilinx Zynq-7000 family opens complete new opportunities to embedded system developers. The two main parts of the system-on-chip, the **Processing System (PS)** and the **Programmable Logic (PL)** are independently placed in the silicon. The device's power circuitry is also designed for separate operation, so that both parts can either work together, alone or dynamically switched off. However the main use case normally is by taking the advantages of the combined parts as a powerful embedded system. Figure 3.2 gives an overview of the Zynq general architecture.

Processing System

The Zynq-7000 family common processing system consists of a dual-core ARM Cortex-A9 processor, which can be operated at frequencies up to 1 GHz, depending on the device. Beside the ARM there are several additional associated processing resources.

[2]https://www.xilinx.com/products/design-tools/vivado.html

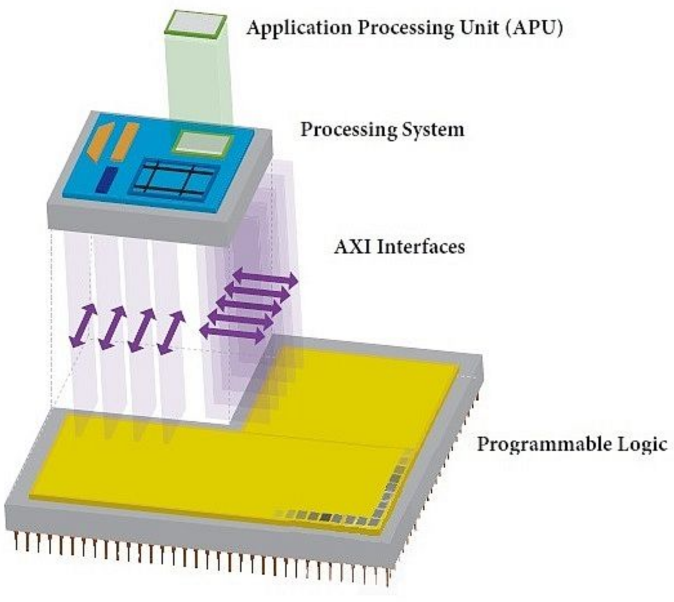


Figure 3.2: Zynq general architecture [CEES14]

- NEON Media Processing Engine (MPE) for Single Instruction Multiple Data (SIMD) processing
- Floating Point Unit (FPU)
- Memory Management Unit (MMU)
- 32kB Level 1 and 512kB Level 2 caches
- 256kB On Chip Memory (OCM)
- Snoop Control Unit (SCU)

All these components together with the two ARM cores form the **Application Processing Unit (APU)**. Furthermore the APU is connected to several peripherals as SPI, I2C, UART, CAN, USB, Gigabit Ethernet and GPIOs. Also different memory interfaces such as QSPI flash, NAND/NOR flash, SRAM and SD card are supported. These peripheral and memory interfaces can be freely activated and assigned to the MIO-pins of the PS, which offers a high flexibility to the user. Figure 3.3 gives an overview of the Zynq processing system.

Up to **54 peripheral signals** can be routed to the according MIO package pins. If more PS-peripherals would be necessary, up to **138 additional signals** can be routed through the PL via the EMIOs and assigned to free pins of the PL part. As calculated in subsection 3.3.1 the platform must have at least 176 free input pins for the PL, but if the PS input can meet the high-speed signal requirements, they could be distributed to PS and PL as well. This has to be considered when selecting the appropriate device-package combination.

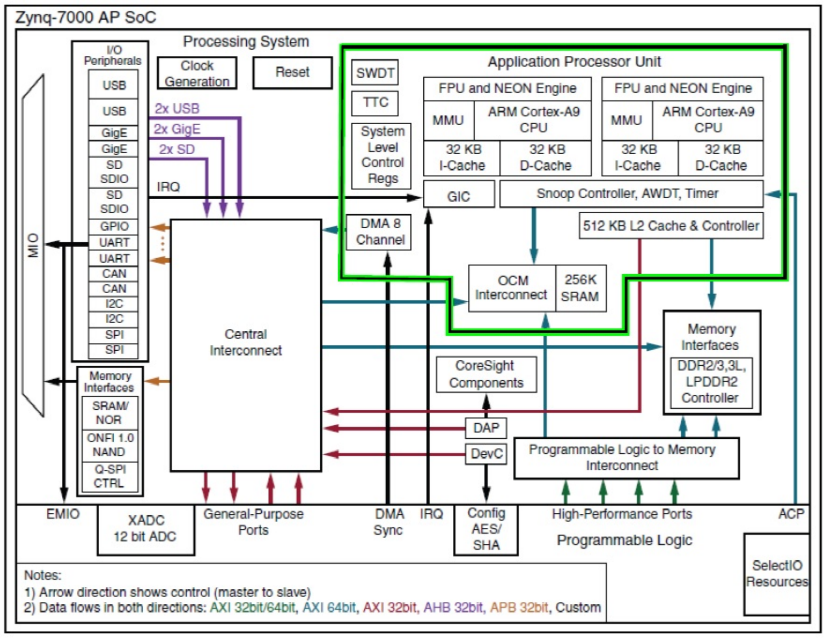


Figure 3.3: Zynq Processing System [CEES14]

A further important feature of the PS are the available timers. They are necessary for the rise time check which is described in detail in subsection 3.4.3. The Zynq device features a 64-bit global timer, which is shared across the two ARM cores. At system level there is a 24-bit watchdog timer and **two 16-bit triple timer/counters**. The latter two are always clocked at 1/4 or 1/6 of the CPU frequency [Xil16b], which depends on the device and the speed grade (SG).

Device	*Frequency [MHz]*		
	SG -1	SG -2	SG -3
Z-7007S/Z-7012S/Z-7014S	667	766	n.a.
Z-7010/Z-7015/Z-7020	667	766	866
Z-7030/Z-7035/Z-7045	667	800	1000
Z-7100	667	800	n.a.

Table 3.4: Zynq PS operating frequencies (n.a.: not available)

When taking the lowest operation frequency and assuming a timer clock at 1/6 CPU frequency, the average timer clock period can be calculated:

$$\frac{667\,\text{MHz}}{6} = 111,167\,\text{MHz} \rightarrow \frac{1}{111,167\,\text{MHz}} = 8,996\,\text{ns} \tag{3.6}$$

By configuring the timer interrupts of the PS, this timer clock period means an interrupt rate of approximately 9 ns. As the specifications in section 3.1 define, the rise time of the print-heads' fire-pulse signal is between 0,5 and 2 μs. To be able to measure the occurring rise time in a sufficient resolution, a **timer interrupt rate of at least 10 ns** is necessary, which can be fulfilled with the calculated solution.

Programmable Logic

The PL part of the Zynq is based on Xilinx 7-series Artix FPGAs for cost-optimized devices and on Kintex FPGAs for mid-range devices. The main features of the Artix FPGA fabric can be reviewed in section 3.3.2.

Just as for the first platform approach, the PL part of the Zynq device also has to fulfill the hardware requirements as in subsection 3.3.1. In Table 3.5 the possible devices and properties are listed. Here the same logic cell density above 35.000 is considered for device selection.

Device	*Package*	*Logic Cells*	*GTP modules*	*PL I/Os*	*FPGA fabric*
XC7Z012S	CLG485	55.000	4	150	Artix-7
XC7Z015	CLG485	74.000	4	150	Artix-7
XC7Z030	CLG485	125.000	4	150	Kintex-7
	FBG676	125.000	4	250	Kintex-7

Table 3.5: Zynq PL device parameters

When looking at the logic cells and GTP numbers all devices meet the requirements. Another aspect are the available PL I/Os. If all necessary signals should be directly connected to PL fabric pins, only the XC7Z030 with 250 available I/Os is suitable. But as previously mentioned here also the high-speed PS-MIOs could be used for certain signals, so that probably also the **XC7Z012S** is suitable.

3.4 Design considerations

For the final platform selection some further considerations have to be made. The most important parameter here is the possible HW/SW separation. This can only be made by analyzing the required core functions of the test procedure.

3.4.1 Data lines test

The test for the data lines is mainly focused on correct signal transmission. As calculated in subsection 3.2.1 data rates of up to **24 Mbit/s** have to be processed by the processing platform. The tough timing constraints of this subtest require real time processing of the incoming signals. Furthermore appropriate interrupt routines have to be implemented when placing this subtest in the PS.

Another criterion is the amount of necessary input pins of the device. Each test target has **128 data lines** that have to be verified. Therefore the same number of either PS-GPIOs or PL-pins must be available. Additionally the data line test has to check **24 control signals** which directly correspond to the data lines. Hence 24 more input pins are needed. If the pin count would get critically due to that, some of the control signals could be routed over a multiplexer to a smaller number of input pins.

3.4.2 Amplitude test

For the amplitude measurement of the PZTDrive signal basically ADCs are used. They also have to be capable of high speeds. The necessary sample rate highly depends on the defined fire-pulse signal shape. Normally this signal is a **multi-pulse series** for precise drop size jetting as seen in Figure 3.4. The maximum printing frequency of the Samba print-head is 100 MHz as described in subsection 3.2.1. In the current printing system these

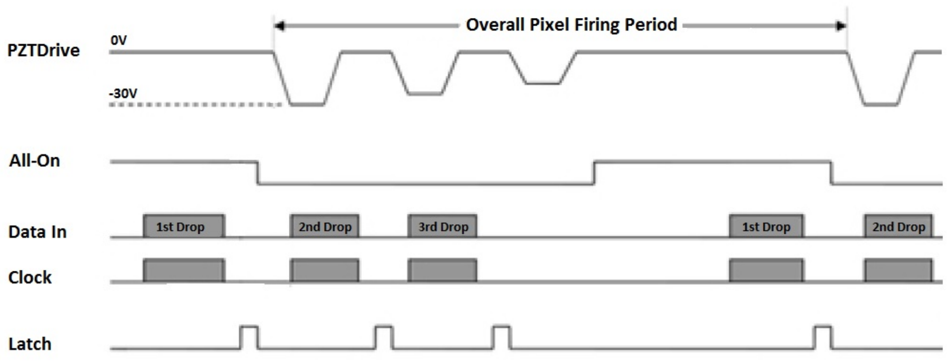


Figure 3.4: Samba analog output signals [FUJ16]

pulses have an overall frequency of about 65 kHz (15 μs period), whereas the high level of one fire-pulse is approximately **1** μ**s** long. By following the Nyquist-Shannon sampling theorem the minimal sample rate of the ADC has to be at least twice the highest signal frequency, which is in this case 500 kHz. Thus the real sample rate should be not less than **1 MSPS** to detect the amplitude in this short high level phase.

3.4.3 Rise time test

The specifications in section 3.1 define an average rise time between 0,5 and 2 μs for the PZTDrive signal. This means a much higher sample rate as for the amplitude test. Before a meaningful proposal of possible hardware components can be made, a closer look at the appropriate mathematical basics of linear functions should be done.

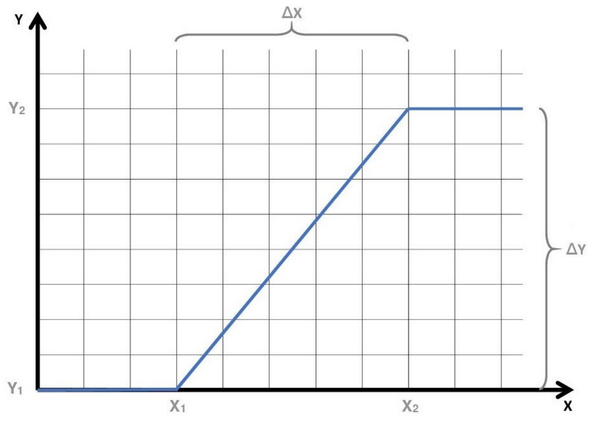
The gradient k of an ideal rectangular pulse is defined as infinite, whereas for a real pulse $k \in \mathbb{R}$. When taking the general form of the straight line equation and assuming $d = 0$, the gradient k can be defined as delta y by delta x.

$$f(x) = k \cdot x + d,\ d = 0 \rightarrow k = \frac{f(x)}{x} = \frac{\Delta y}{\Delta x} \tag{3.7}$$

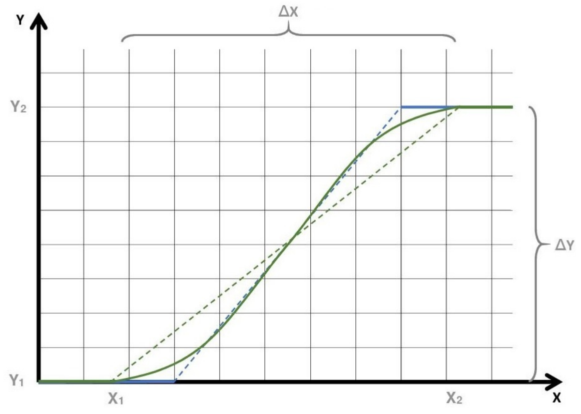
Furthermore Δy can be defined as the difference of y_2 and y_1. By deploying this and the same for Δx in Equation 3.8 the gradient k can be calculated as following.

$$k = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.8}$$

With this correlation the rise time can be simply calculated as the gradient of the signal's rising edge. The two different points for each axis can be freely defined when assuming ideal linear slopes as in Figure 3.5a. In a real life electronics environment the signal slope will more look like in Figure 3.5b. In this case the real slope is more rounded off near the signal's minimum and maximum. With interpolating the linear part of the real slope the overall rise time can be calculated. If the difference is not too big, the resulting rise time error could be ignored. But because this it is not absolutely deterministic due to electrical characteristics like parasitic effects and component tolerances, a better solution than ignoring should be found.



(a) Ideal linear slope



(b) Real slope

Figure 3.5: Different signal slopes for rise time calculation

A probably more sophisticated approach for this is to consider only the linear section of the real slope as in Figure 3.6. This can be achieved by defining two designated thresholds for the y-axis, which limits the measurement to the desired part of the rising edge. Thus the rounded signal parts cannot distort the rise time calculation.

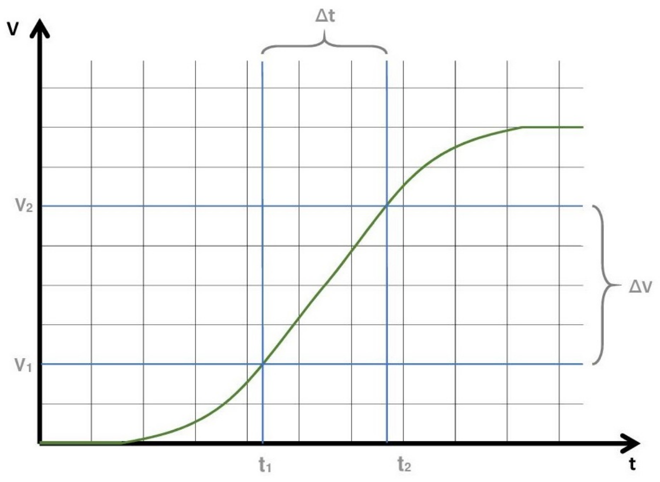


Figure 3.6: Linear region of the rising edge

Before processing the rise time measurement an according hardware circuit has to be designed. This was already done in [Hal17] and can be seen in Figure 3.7. The PZTDrive signal is first scaled down by an inverter circuit to a range of 0-3 V. Then this scaled down signal is fed into a comparator circuit, where the reference voltages are predefined to the mentioned threshold voltages. This results in two converter output signals, which are basically rising edges at a certain point in time. Due to the different timestamps of the output pulses, the processing platform can calculate the rise time out of the time difference and the known threshold values.

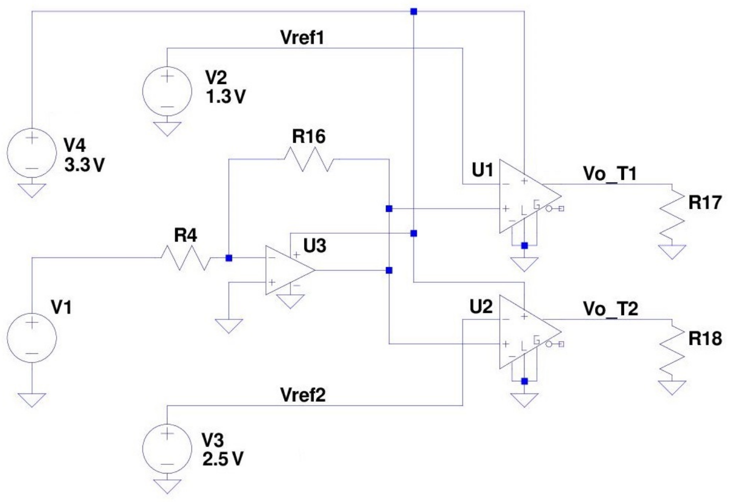


Figure 3.7: LTSpice rise time circuit

When taking the **maximum slew rate** of about **30 V/500 ns** and assuming that due to the previously mentioned round-off errors only half of the rising edge can be used for calculation, the effective measurement time for the linear part is limited to approximately 250 ns (Equation 3.9 and 3.10). This means that the processing platform has to handle **input frequencies up to 4 MHz**.

$$\frac{30\,\mathrm{V}}{500\,\mathrm{ns}} = 60\,\mathrm{mV/ns} = k \tag{3.9}$$

$$V(t) = k \cdot t \rightarrow \Delta V = k \cdot \Delta t \rightarrow \Delta t = \frac{15\,\mathrm{V}}{60\,\mathrm{mV/ns}} = 250\,\mathrm{ns} \tag{3.10}$$

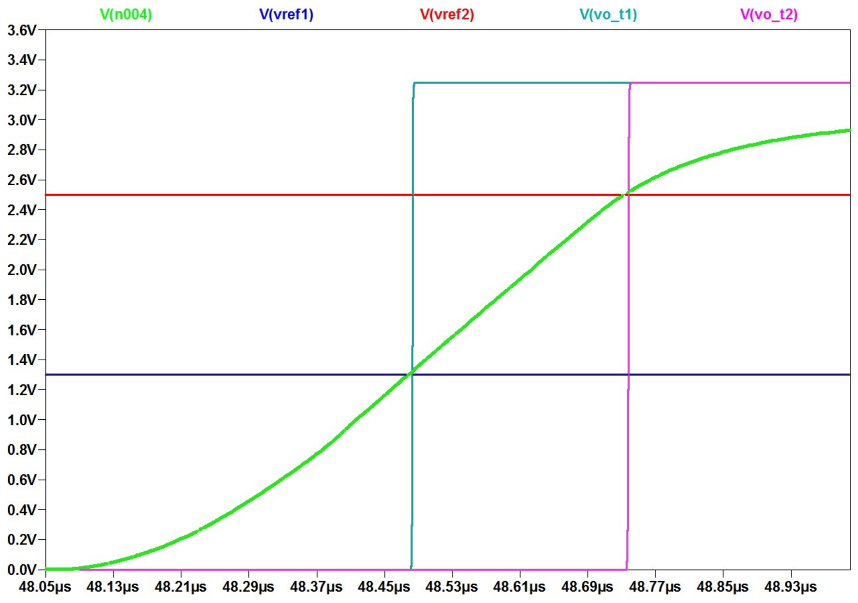
In the real printing environment the waveform for PZTDrive is generated by a power amplifier stage on the control electronics. This driver stage was also created as an appropriate LTSpice simulation circuit. The output waveform of the simulation can be exported as text file and used for other circuits as input signal. In this case the **rise time is defined with 640 ns**. For the schematic in Figure 3.7 the source V_1 takes this file as input argument for the fire-pulse generation. The inverter stage is configured for a gain of $\frac{1}{10}$. The resulting signal shape can be zoomed to the rising edge as in Figure 3.8. To measure the correct timestamps of the comparator output signals the marker tools were set to the point where the reference voltage levels cross the input signal. The highlighted field of the marker window shows the time difference of the two comparator signals. Out of this and the known threshold levels the effective rise time can be calculated.

$$\frac{\Delta V}{\Delta t} = \frac{(2,5\,\mathrm{V} - 1,3\,\mathrm{V})}{254,85\,\mathrm{ns}} = 4,71\,\mathrm{mV/ns} \tag{3.11}$$

$$\frac{4,71\,\mathrm{mV/ns}}{\frac{1}{10}} = 47,1\,\mathrm{mV/ns} \rightarrow \frac{30\,\mathrm{V}}{47,1\,\mathrm{mV/ns}} = 636,9\,\mathrm{ns} \tag{3.12}$$

$$t_{error} = \Delta t_{rise} = 640\,\mathrm{ns} - 636,9\,\mathrm{ns} = 3,1\,\mathrm{ns} \tag{3.13}$$

The resulting 4,71 mV/ns have to be divided by the gain of the inverter circuit to get the PZTDrive signal rise time as in Equation 3.12. When comparing the predefined rise time with the calculated one, only a **difference of 3,1 ns** occurs. This error value is definitely in the required tolerance as discussed in subsection 3.3.3, where a minimum timer interrupt rate of 10 ns is defined.



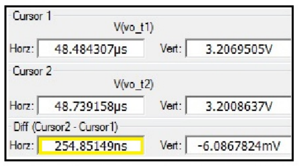

Cursor 1
V(vo_t1)
Horz: 48.484307µs Vert: 3.2069505V
Cursor 2
V(vo_t2)
Horz: 48.739158µs Vert: 3.2008637V
Diff (Cursor2 - Cursor1)
Horz: 254.85149ns Vert: -6.0867824mV

Figure 3.8: LTSpice rise time simulation results

As a conclusion of this calculations can be said that an embedded processor has to feature configurable interrupts for GPIO pins at a speed of at least one interrupt every 10 ns. Due to the fact that overall 8 PZTDrive signals have to be tested and each measurement needs 2 comparator signals, a total of **16 dedicated I/Os with configurable interrupts** is necessary for the rise time tests.

3.4.4 HW/SW separation

The HW/SW partitioning is a key element in the typical embedded system design flow. Figure 3.9 shows a typical representation of the design flow. One can clearly see that

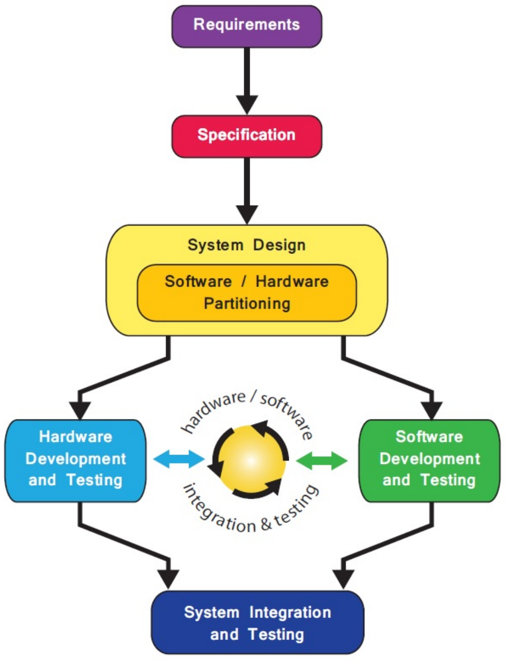

Requirements
Specification
System Design
Software / Hardware Partitioning
Hardware Development and Testing
hardware / software
integration & testing
Software Development and Testing
System Integration and Testing

Figure 3.9: Typical model of a HW/SW design flow [CEES14]

the partitioning phase is probably the most important step in the whole design process, because it has major influence for all further implementation iterations.

As already discussed, the best fitting platform of the proposals in subsection 3.3.2 and 3.3.3 must be chosen according to the calculated parameters. Furthermore the placement of the appropriate modules for all required functions is influencing this decision. With the results of the previous sections, the separation of hardware and software parts should be generally possible, but first certain functional modules have to be identified based on the defined requirements.

Test procedure control logic: This is the main data flow control unit which handles the overall communication between all the special function modules.

Host PC communication: A host PC is used to generate and transmit the test data for certain test cases. Furthermore the test results and also debug information should be send back via this module.

Test target communication: The preprocessed test data has to be transmitted to the test target PCB over a high-speed GTP-interface.

Amplitude data acquisition: For gathering the analog input signals, this module has to handle the SPI-communication and data acquisition with certain ADCs.

Data line signal integrity check: The test target processes the appropriate nozzle data for the print-heads. This module should handle the data comparison and correct timing of this received data signals.

PZTDrive amplitude check: The amplitude input data from the ADC-SPI module has to be processed and verified.

PZTDrive rise time check: As described in subsection 3.4.3 this module has to implement the rise time check with a certain time difference measurement logic.

Next a suggestive distribution of these functional modules have to be found. Therefore the modules are filled in Table 3.6 according to possible ease of implementation in the respective part. This general separation can be analyzed in more detail when looking

	Artix-7 with MicroBlaze		*Zynq-7000*	
Functional module	Hardware	Software	Hardware	Software
Test procedure control logic		x		x
Host PC communication	x			x
Test target communication	x		x	
Amplitude data acquisition	x			x
Data line signal integrity check	x		x	
PZTDrive amplitude check		x		x
PZTDrive rise time check	x			x

Table 3.6: Possible distribution of functional modules in hardware and software

at the specific parameters of the modules. Especially the limited I/O pin count is a major criterion for the module assignment. The Artix-7 solution with embedded soft-processor core has no dedicated pins for the MicroBlaze, whereas all necessary pins have to be assigned and routed in the FPGA fabric. As mentioned in subsection 3.3.2, all of the proposed devices feature enough I/O pins for the required functionality. Therefore the needed I/Os in Table 3.7 are considered only for the second solution with the Zynq SoC. As one may observe that for some modules like the GTP, which are proposed to be

		Zynq-7000	
Functional module	*Purpose*	PL I/O pins	PS MIO pins
Test procedure control logic	Status LEDs	0	1
Host PC communication	UART, USB	0	6
Test target communication	GTP I2C and control	0	3
Amplitude data acquisition	SPI for 8x ADCs	0	6
Data line signal integrity check	Data-in and control	145 (152)	3 (0)
PZTDrive amplitude check	Internal module	-	-
PZTDrive rise time check	Comparator outputs	0	16
	Total pin count	145	35

Table 3.7: I/O pin count of the specific modules on the Zynq-7000 platform

implemented in HW, no PL I/O pins are counted. This is because the GTP has dedicated pins for the module in each device package so that only 3 signals for the control and configuration interface are left. Another point to mention is the SPI interface for the amplitude test ADCs, where 8 single channel converters share the same SPI data and clock lines. The other 4 signals are used for a 3-to-8 decoder for the chip select signals of each ADC. Furthermore when looking at the data-in signals, there are 128 in total plus 24 control lines. Since the most of the mid-range Zynq devices feature only 150 PL I/Os as in Table 3.5, a multiplexer IC could be used for the 8 AllOn signals. Then only 145 PL pins and additionally 3 PS MIOs for the signal selection logic would be necessary.

3.4.5 Platform selection

With all the knowledge gained during the design phase it could be proven that both solutions can fulfill all necessary requirements. To summarize the gathered information, the pros and cons of both platforms are listed as follows.

Aritix-7 with MicroBlaze

Advantages:

- Already well proven platform within the development department with good know-how
- High number of available PL I/Os
- MicroBlaze soft-processor cores can be instantiated and configured in the FPGA fabric as needed
- Probably easier HW design due to missing external DDR memory

Disadvantages:

- Much less processing power due to embedded soft-processor core
- More modules are necessary to be implemented in hardware as a consequence of using the MicroBlaze
- Higher HW development effort due to necessary integration of IP-cores for basic communication functionality as I2C or SPI interfaces

Zynq-7000

Advantages:

- High processing power with dedicated dual-core ARM processors on the SoC
- More modules capable to be implemented in software running on the PS
- Faster development due to higher number of SW-parts with reusable code libraries
- Better suitable platform for future applications with rising degree of complexity
- Possibility to add MicroBlaze co-processor cores in PL for even more complex applications

Disadvantages:

- Higher development effort due to new device architecture with lack of know-how
- More additional parts like external DDR memory needed
- Higher amount of PCB development due to the external memory

Conclusion

The probably most important advantage of the Zynq SoC is the high processing power of the dedicated ARM Cortex-A9 hard-block processor. As seen in the HW/SW separation in subsection 3.4.4 a lot of functional modules can be placed in the software site of the Zynq device because of the better performance and processing speed. Also the already implemented software libraries for a lot of the necessary functions decrease the development time. The disadvantage of the higher effort due to lack of know-how is on the other hand a compromise with gaining new experience for future applications.

To come to a conclusion, the **XC7Z015** device from the Xilinx Zynq-7000 family is probably the better fitting platform in this case. It features enough I/Os in the Artix-7-based PL for the high-speed data acquisition while also being able to handle the PS related data input signals. Other than the XC7Z012S device with only one ARM core the XC7Z015 with its dual core processor is a very powerful but also flexible platform to be ready for more advanced future tasks.

3.4.6 Zynq System-on-Modules

With the evaluation and selection of the right platform for this work, now an appropriate infrastructure to develop the system application on a real device has to be chosen. Before designing and producing a dedicated printed circuit board, the usual design flow is to try out the basic functionality as a proof-of-concept on an existing hardware platform. Within embedded system design a common way for this are evaluation boards, which are normally provided by the SoC manufacturer or third-party suppliers. The major drawback of such eval-boards is that they are not customizable. Typically they feature every possible peripheral of the SoC available on the board to show it's strength, whereas valuable I/Os could be used for more necessary functions. Another aspect is that such boards are normally not designed for integration into field tests or product enclosures as they have neither the robustness nor form factor for this [Joh15]. Nevertheless when it comes to the development of the first prototype, a lot of know-how and experience is necessary to design proper PCBs, especially in high frequency mixed signal domains.

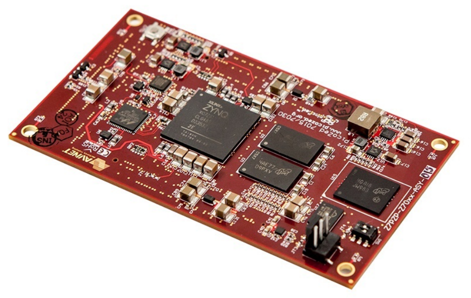
Therefore semiconductor companies and their partners usually provide reference designs with already certified matching and power supply circuits, so called **System-on-Modules (SoM)**. The main benefit for the customer is that he can take a ready-to-use board design with proven functionality and saves a lot of time and costs for the first prototyping phase. Furthermore the developer has still full flexibility of how to use the available peripheral I/Os best for his needs. Beside that customers must not necessarily have the know-how and experience for such complex PCB designs. They only have to develop a carrier- or interface-board for the SoM, which can be already designed for system integration. In some cases it might also be valuable to directly design-in the SoM into the final product since they normally have a good availability and multiple times lower costs at higher volumes.

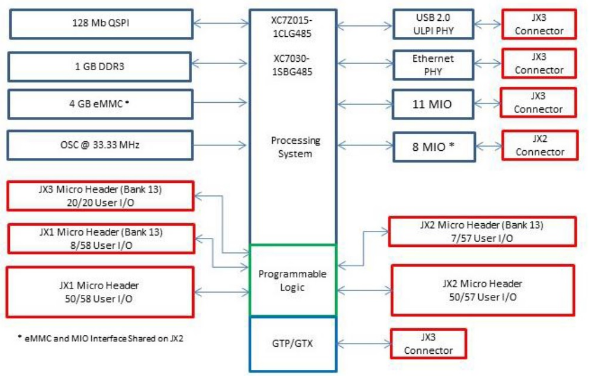
For the Xilinx Zynq device several different System-on-Modules are available from different manufacturers. On the Xilinx website [Xil17a] one can search for all Xilinx-certified partner products. Another useful and independently constituted comparison of Zynq-SoM boards [Joh15] show up the most popular and versatile starting points for Zynq PCB development.

During the design phase of this work also several possibilities were discussed since it figured out that an appropriate Zynq SoM with carrier card should be the best solution for a PoC. One important reason for this was that some companies offer a broad range of SoMs with normalized form factors for different processing platforms such as Artix-7 FPGAs and Zynq-7000 SoCs. This means that the carrier card only has to feature the same footprint as the SoMs, so that the system integration can be simplified by just changing the System-on-Module. In this case also an easy comparison of different platforms can be made with very low effort. The company Trenz Electronic offers a broad range of different FPGA and SoC modules with pin-compatible footprint such as the TE0715[3] or TE0711-01-35-2I[4] SoM.

PicoZed SoM

With the comparison in [Avn17] and further research finally the very popular **PicoZed**[5] board with the appropriate **carrier card**[6] from Avnet was chosen for the first PoC. The


(a) PicoZed 7015



(b) Block diagram

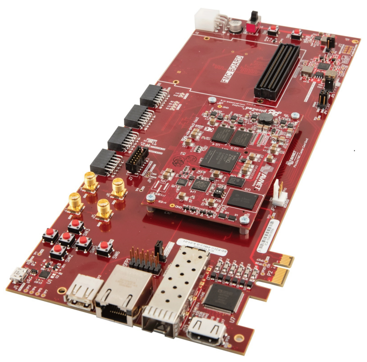
Figure 3.10: PicoZed 7015 SoM with overall block diagram

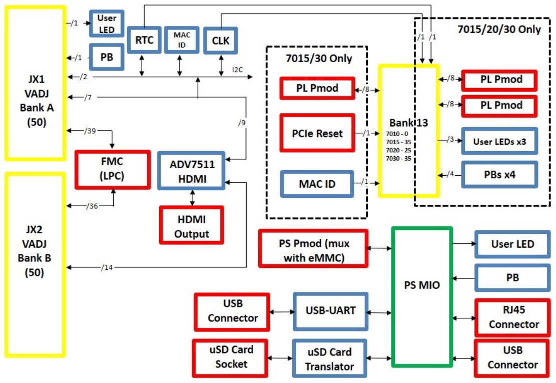
PicoZed is also designed with a footprint-compatible form factor featuring the Xilinx Zynq-7010, 7015, 7020 and 7030 devices. The 7Z015 version embeds the Zynq-7015 SoC and fulfills the most of the requirements for this work, more about this in later. Table 3.8 shows the main features of the used module and in Figure 3.10b the overall block diagram can be seen. On the backside of the module there are three I/O connectors, which provide easy carrier card access to the majority of the Zynq I/Os, the GTP transceivers, USB, Ethernet and JTAG. The counterparts of this connectors are available on the PicoZed FMC Carrier Card V2 which embeds the appropriate PHYs and interfaces for implementing the PicoZed's features. Furthermore the complete power supply circuity is also on-board of the carrier card, with allows a plug-and-play starting point for prototyping.

[3]https://shop.trenz-electronic.de/de/Produkte/Trenz-Electronic/TE07XX-Zynq-SoC
[4]https://shop.trenz-electronic.de/de/TE0711-01-35-2I-High-IO-Industrial-grade-Xilinx-Artix-7-35T-Module-with-speedgrade-2I-and-USB?c=140
[5]http://picozed.org/product/picozed
[6]http://picozed.org/product/picozed-fmc-carrier-card-v2

Features	*PicoZed 7Z015*
SoC	XC7Z015-1SBG485
User I/Os	148 User I/O (135 PL, 13 PS MIO)
Memory	1 GB of DDR3 SDRAM
	128 Mb of QSPI Flash
	4 GB eMMC
high-speed transceiver	4x GTP

Table 3.8: Main features of the PicoZed 7Z015 SoM


(a) FMC Carrier Card V2



(b) Block diagram

Figure 3.11: PicoZed FMC Carrier Card V2 with overall block design

Drawbacks

As some might have noticed, the PicoZed actually does not provide the necessary amount of I/Os for implementing all of the required test cases. It also has not the necessary ADC and comparator circuits for the tests in section 3.4 implemented. The reason why this board anyway has been chosen is that no better SoM alternative with matching I/O count and support of GTP transceivers currently is available. Nevertheless the whole system provides enough features to implement the core functionality for the final data acquisition task:

Basic control logic: The basic control functionality is independent from the overall hardware infrastructure. Therefore later on it should be relatively straight forward to adapt the additional functionality.

Host communication: The USB and UART software modules also can be pre-programmed on any similar platform due to a common SW-library.

Test target communication: The PicoZed carrier card features one SFP cage for one GTP channel to simulate the target communication functionality in loopback mode. This can also be combined with the data line signal integrity test. Instead of checking single I/O pins, the received high-speed data from the transceiver can be used to verify the data.

Rise time test Instead of gathering external I/O data from the PS MIO pins the Zynq offers the possibility to route the PS GPIOs into the PL. With appropriately generated signal rising edges at proper timestamps in the PL, the PS EMIOs can also raise edge triggered interrupts for the rise time test.

The rest of the defined modules in subsection 3.4.4 can be easily adopted later on the final hardware platform, which is discussed in the following section.

3.5 Test module PCB

For a first proof-of-concept in the early prototyping phase, the PicoZed System-on-Module is a perfect hardware platform. Anyway the final project goal is to have a dedicated printed circuit board with all required functionality for the print-head control electronics verification. With the FPGA evaluation and platform selection already discussed, the Xilinx Zynq XC7Z015 SoC can now be integrated into an embedded system PCB design. This already was done during a seminar project as one can see in detail in [Hal17]. For the test module PCB there are the following hardware specifications stated.

3.5.1 PCB specifications

1. The most important signals to test are the 2x16 data lines per print-head. These signals are connected via two flex cables to the control PCB. Each single data line has also to be connected to the processing system.
2. Beside the data lines also the signals CLK, Latch and AllOn for controlling the print-head ASICs should be verified and connected.
3. The fire-pulse voltage for the print-heads is generated by the driver stage of the control PCB with a special voltage potential design. The test module should implement a proper circuit to transform the voltages to an applicable range with respect to the board GND.
4. Each print-head has two on-board EEPROMs for configuration and a thermistor for safety issues. The data of these devices should also be read out and verified.
5. For the high-speed data transfer a fiber optical interface should be used
6. To be able to communicate with a host computer an RS485 interface should be connected to a UART port of the processing system.
7. A USB-connector should be available for debug capabilities.
8. The power supply should be compatible to the control PCB with 48 VDC.

3.5.2 Overall block design

With the Zynq SoC as central processing platform one can see the main hardware parts in the overall block diagram in Figure 3.12. The power supply circuit is leaned on common FPGA and SoC supply schemes, but adopted according to the special floating voltage

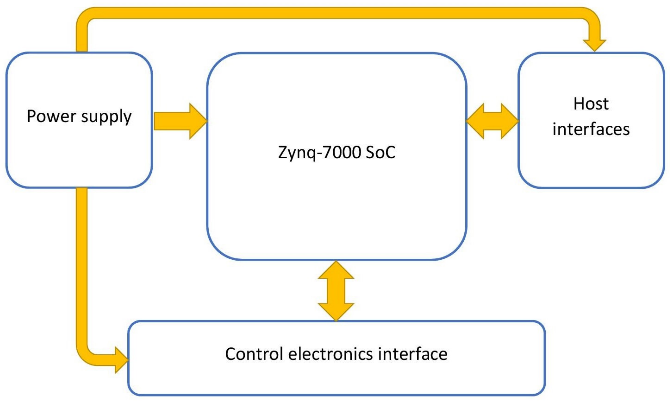


Figure 3.12: Overall block design of the test module [Hal17]

design of the print-head signals. A flyback converter with multi-point transformer is used to generate the supply and reference voltages for the ADC and comparator logic. For the Zynq power supply a multichannel buck DC/DC converter is used with properly defined power up sequence. Special attention was given to a certain Zynq security feature called Secure Lockdown[7], where the power up sequence of all SoC supply voltages has to meet specific timing constraints.

To interact with a host PC two UART interface PHYs are used to convert the PS UART signals to RS485 and USB protocols. Additionally the Zynq JTAG interface is routed to an appropriate pin header, which is compatible with common Xilinx debug probes. The control electronics interface is divided into two parts, the fiber-optic cable (FOC) transmitter and the flex-cable circuit receiver. These interfaces build the main test target data flow ring.

3.5.3 Dataflow

The test module PCB is designed to work in an industrial test environment, where the whole test procedure should not produce a huge time overhead. Therefore a host PC is connected to the module to easily transmit the test data and to start the test run. The rest of the data processing is handled by the module electronics. In Figure 3.13 the basic dataflow of the module is illustrated.

First of all the user has to define and transmit the test data from the host control PC to the test module PCB. This can be simply done with any terminal program which can send text data. When the Zynq PS has received the data, it is temporarily stored in the PS memory. Right after that the firmware hands over the data to a SW module, which stores the test data into a certain BRAM in the PL. With the appropriate command from the host PC the user can start the test procedure. Now the whole data in the BRAM is sent by the gigabit transceiver of the Zynq over the fiber-optical cable to the test target. There the print-head control electronics processes the data as it normally does with image data. The 2x16 nozzle signals per print-head and certain control signals are then transmitted back to the test module PCB over a flex-circuit cable. An appropriate comparator circuit

[7]https://www.xilinx.com/support/answers/63149.html

generates the signal-pulses for the rise time test. Furthermore multiple ADCs measure the analog amplitude of the PTZDrive signals and transfer the result via SPI to the PS. Special digital isolator ICs are used to transform the signals to the module potential. After that the normalized nozzle-data is send directly to the PL. The certain hard- and software modules in the Zynq process and verify the target data according to the test specifications. Finally the test results are sent back to the host PC.

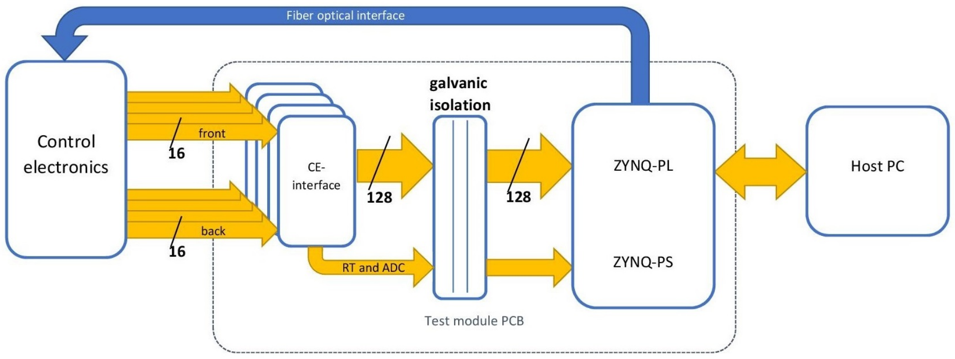


Figure 3.13: Simplified test module PCB data flow diagram [Hal17]

The whole architecture is designed to support also parallel test procedures for multiple modules. Each PCB has a configurable identification numbler to be able of building up a ring network of several test module PCBs. In this case the user can control multiple test runs with a single central host interface. The test data is first transferred to one PCB in the network. With a certain command from the host, this module then distributes the data over the fiber-optical ring.

Chapter 4

Implementation

4.1 PicoZed platform capabilities

For the implementation of high-speed data acquisition and processing for digital print-head systems the PicoZed SoM is used in the first prototyping phase. This actually limits the realizable amount of functionality due to the platform characteristics, but offers a quick and easy function verification. Especially for the implementation of the core modules like the basic control logic on the Zynq SoC it is a great opportunity to check the performance of the system in an early implementation stage. Before the dedicated realization of the particular modules is described in detail, a closer look to the possibilities of the PicoZed SoM and the corresponding carrier card should be made.

4.1.1 PicoZed key facts

Host interfaces For the communication with a host PC the carrier card features a UART-to-USB converter PHY. With this interface the test and result data transmission can be done with an appropriate terminal application on the host PC. During the development phase the on-board JTAG connector can be used for general debug purposes within the Xilinx Vivado Design Suite.

Fiber-optical interface The carrier card also has an SFP-cage for the FOC-interface on board which is directly connected to one of the four gigabit transceivers of the PicoZed's Zynq SoC. With this interface the target communication can be simulated and verified in a loopback configuration.

User I/Os The majority of the Zynq's I/O lines are routed to the carrier card via three connectors. 30 of the PL I/Os and 10 PS I/Os are available on Pmod connectors. The rest is internally assigned to several peripherals such as 5 PL push buttons and one for the PS. These push buttons can be used to implement the GPIO interrupts which are needed for the rise time test. Also in total 6 user LEDs can be used for status visualization.

QSPI Flash For the Zynq boot procedure a non volatile memory is needed to store the FSBL and the application code. Therefore the PicoZed has a 128 Mbit NOR flash with 4-bit SPI interface on board. The SPI lines are connected to MIO1-8 and shared with the Zynq bootstrap pins for boot-mode selection.

There are also a lot more interfaces and resources available on the PicoZed system such as USB OTG, **Gigabit Ethernet**, PCIe, HDMI and a real time clock IC. They are not needed for the current prototype specification but could be useful for future requirements. Especially the Ethernet could be an alternative for the RS485 host PC communication channel.

4.1.2 Operating systems for Zynq

The Zynq platform is currently configured to run only standalone bare-metal applications, but it is also capable of running operating systems such as RTOS or even **embedded Linux**. Running Linux on the final test module could be interesting for future applications, where the printing infrastructure communication or test data preprocessing could be directly implemented on the test module. This would lower or even eliminate the need of an additional host PC. Additionally with developing device drivers for interfacing between Linux and the particular PL modules, the complete control logic and system communication could be driven within the operating system. All these features together could lower the overall communication overhead and simplify the application development.

Embedded Linux distributions for Zynq

Xilinx offers an open source Linux OS called **Zynq-Linux**[8]. Based on the Linux 3.0 kernel it includes several additions from Xilinx like the BSP and certain device drivers for Timers, Ethernet, GPIO, I2C, SPI etc. which speeds up the development.

Another embedded Linux distribution is provided by Petalogix®. Their product Petalinux is a combination of a fully-functional Linux distribution and an IDE which can be integrated with the Xilinx design flow[9].

The company Wind River provides a certified, secure distribution for the PicoZed under the name **Pulsar™Linux**[10]. According to Wind River, Pulsar is mentioned to speed up embedded and IoT application development with free security updates.

[8]http://www.wiki.xilinx.com/Zynq+Linux
[9]https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html
[10]https://www.windriver.com/products/operating-systems/pulsar/avnet_picozed.html

4.2 Test procedure

Now knowing enough about the capabilities of the hardware platform, the test procedure can be defined. A common way to describe such procedures are UML diagrams. For the description of the whole test procedure different data and control flow representations can be used.

4.2.1 Structural description

To illustrate the overall structure of the test module implementation a UML component diagram is used. Figure 4.1 shows the main components of the final test procedure with all test cases covered. Generally it consists of three modules which have certain communication interfaces. The central unit is the test module with the Zynq PS and PL parts. Additionally it contains the converter and comparator logic components for the particular test cases.

In the Zynq PS part, the **main control logic** for the test procedure is implemented. It is responsible for the overall control flow of all involved components. A **test data buffer** is used to temporarily save the test data from the host PC. Some of the PS MIO pins are configured as **SPI interface**, which is used to gather the digitally converted data from the 8 ADCs. With a chip select signal logic the appropriate ADC can be chosen for the data transmission. Further MIOs are configured as GPIO-inputs with edge sensitive interrupt trigger to capture the comparator signals. The **interrupt logic** component uses the data from the TTC to process the appropriate rise time values.

To interact between the PS and PL part of the Zynq SoC a special **PL-interface** was developed. As one can see in Figure 4.1 the module functions as communication hub for the main control logic and the **test data generation** component. It also transfers the data of the **data line test** component back to the control logic. As the test data from the host PC is only temporarily buffered in the PS, the **TDG-component** takes this data and writes it into a dedicated **BRAM instance**. With the respective command from the control logic the data is read out from the BRAM and send to the **GTP-interface**, which transmits it to the **test target**.

As already discussed in subsection 3.4.5 a proof-of-concept with the PicoZed platform was done, before the adaption to the final implementation on the test module PCB could be realized. Therefore the component diagram has to be modified according to the hardware capabilities. Figure 4.2 shows the adopted component diagram for the current implementation stage. One can clearly see that the grayed out external test target part is replaced by internal components, mainly in the PL part. Furthermore the core components are the same as in the final stage. Especially in the PS part all functional units except the SPI-interface can be adopted one-to-one.

Thus, the major changes are necessary in the PL. Due to the missing test target, the **GTP-interface** is now configured in **loopback mode**. For the signal integrity check all received data bytes are directly transfered to the data line test component within the PL. With this workaround the test target transmission and data line test can be simulated without major modifications of the affected components.

Another required adaption is the data acquisition from the comparators for the rise time test. Without the missing comparator circuit, the respective signals have to be generated alternatively. Therefore an additional component (**CSG**) was developed which generates similar comparator signals directly in the PL part. To still be able to gather the signals with the existing PS components, the **EMIO** functionality is used. This is a Zynq specific possibility to extend the available amount of MIOs by routing them into the PL. Thus, a direct interface for the GPIO interrupts generated by the PL is created.

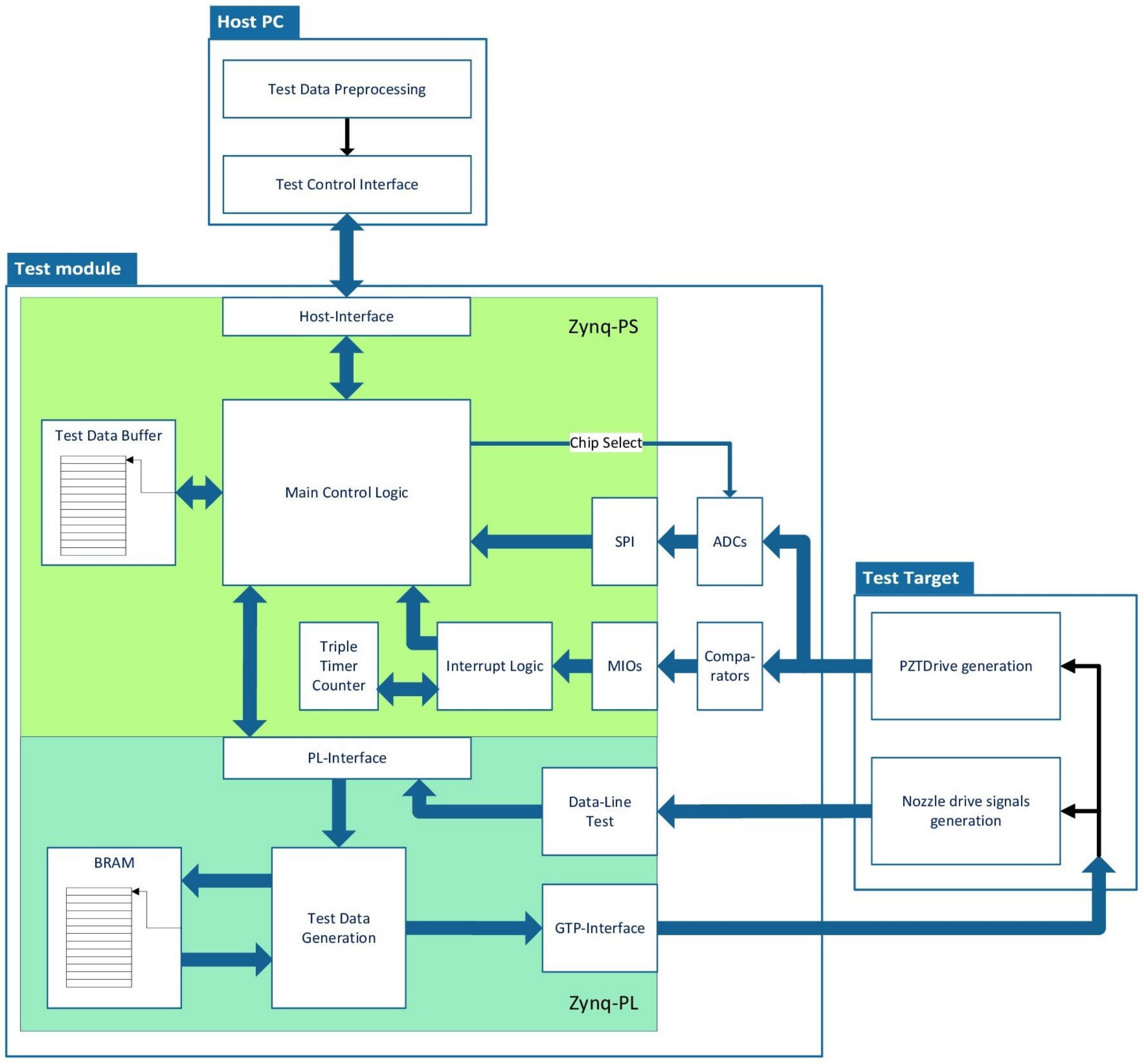


Figure 4.1: Component diagram of the final stage test procedure

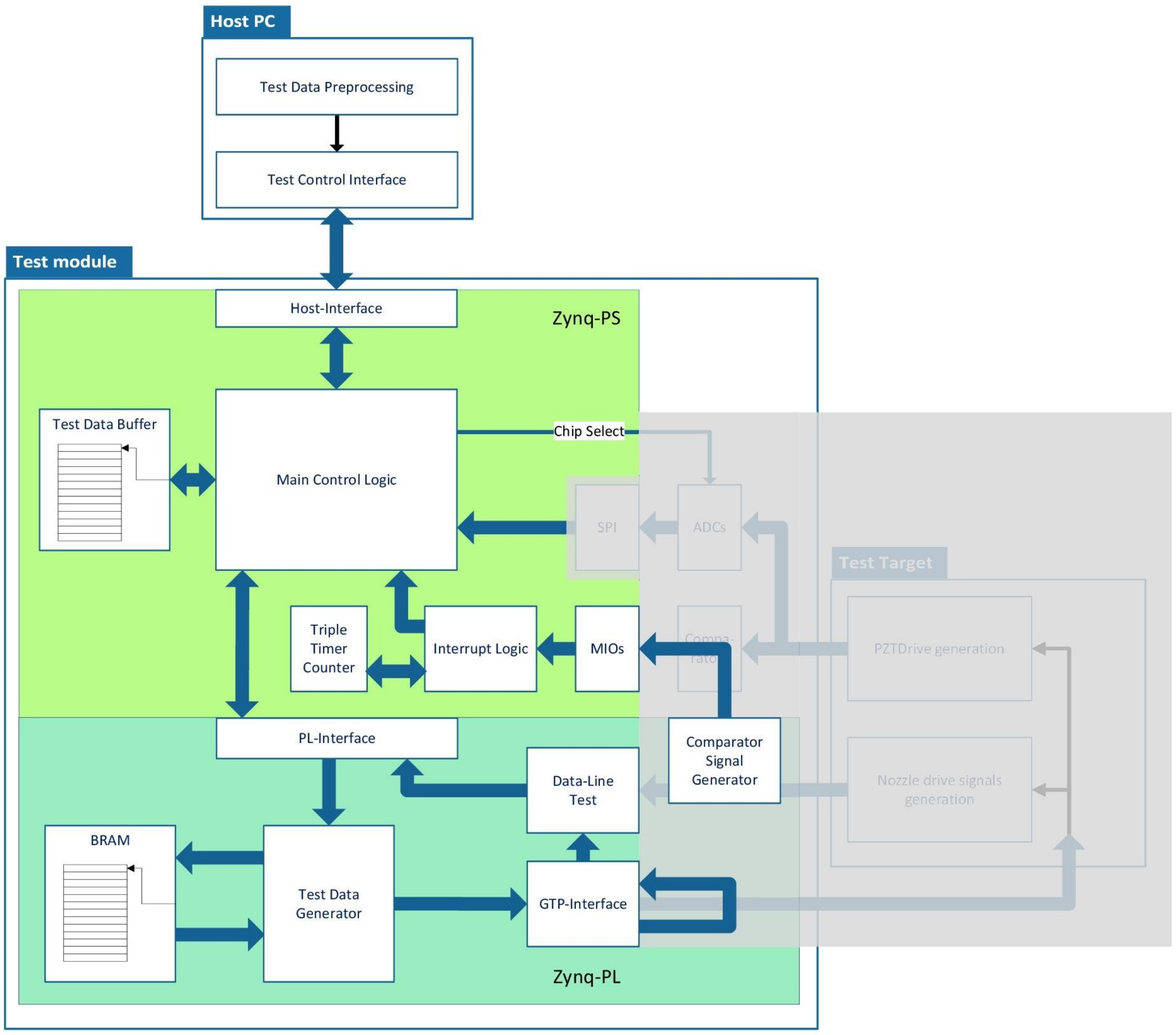


Figure 4.2: Component diagram of the current stage test procedure

4.2.2 Behavioral description

The UML activity diagram as seen in Figure 4.3 is a directed graph similar to a state chart, where all possible states (activities) are represented by nodes and state transitions are denoted by connectors. Parallel activities start and end with horizontal bars.

On the top of the diagram the test data preprocessing is the initial activity and at the same time kind of an idle state for the test module. The user defines there the test pattern for the target on the host PC. With the appropriate command, the next state test data transmission is activated. When the Zynq PS successfully received all data it goes into the buffering state. There several commands are possible to be executed. Normally, during buffering at this point the test data in the PS, the command *write_to_BRAM* is executed. It takes the temporarily saved data and transfers it over the PL-Interface and the TDG into the particular BRAM of the PL. In the case any failure is occurring at this state, the test module requests new test data from the host, which will lead to the initial activity. Otherwise the *start_test* command proceeds onwards.

Here, after sending the test pattern to the target, the Zynq goes into a wait state until the print-head electronics has processed and send back the data. When the transmission from the target failed, the control unit moves back to the buffering activity. There either new data can be requested or the test can be repeated with the actual test pattern. When the transmission was successful, the three test cases are started.

The respective nozzle signals are gathered from the transmission data stream and further on processed to check the signal integrity. The PZTDrive fire-pulse signal is processed by the rise time and amplitude check component in parallel. After the comparator pulse generation the rise time check is executed by the PS. At the same time the analog-to-digital conversion and the amplitude check is performed.

This would be the behavior on the final test module PCB (Figure 4.3). Instead of communicating with the target in this phase, the current implementation on the PicoZed platform only returns the test pattern in a loopback mode to the GTP component as illustrated in Figure 4.4. Also here if any error occurs during loopback communication, the transmission can be repeated or new test data can be requested. When the pattern is received back without errors, the next states proceed with nozzle data gathering and comparator signal generation. After that, the two respective test activities continue with data processing.

When all tests are successfully completed, the results are send back to the host PC. If the test runs terminate with errors, the particular tests can be repeated or new data can be requested. Finally the whole procedure end with the correct transmission of the test results, which can then be evaluated on the host PC.

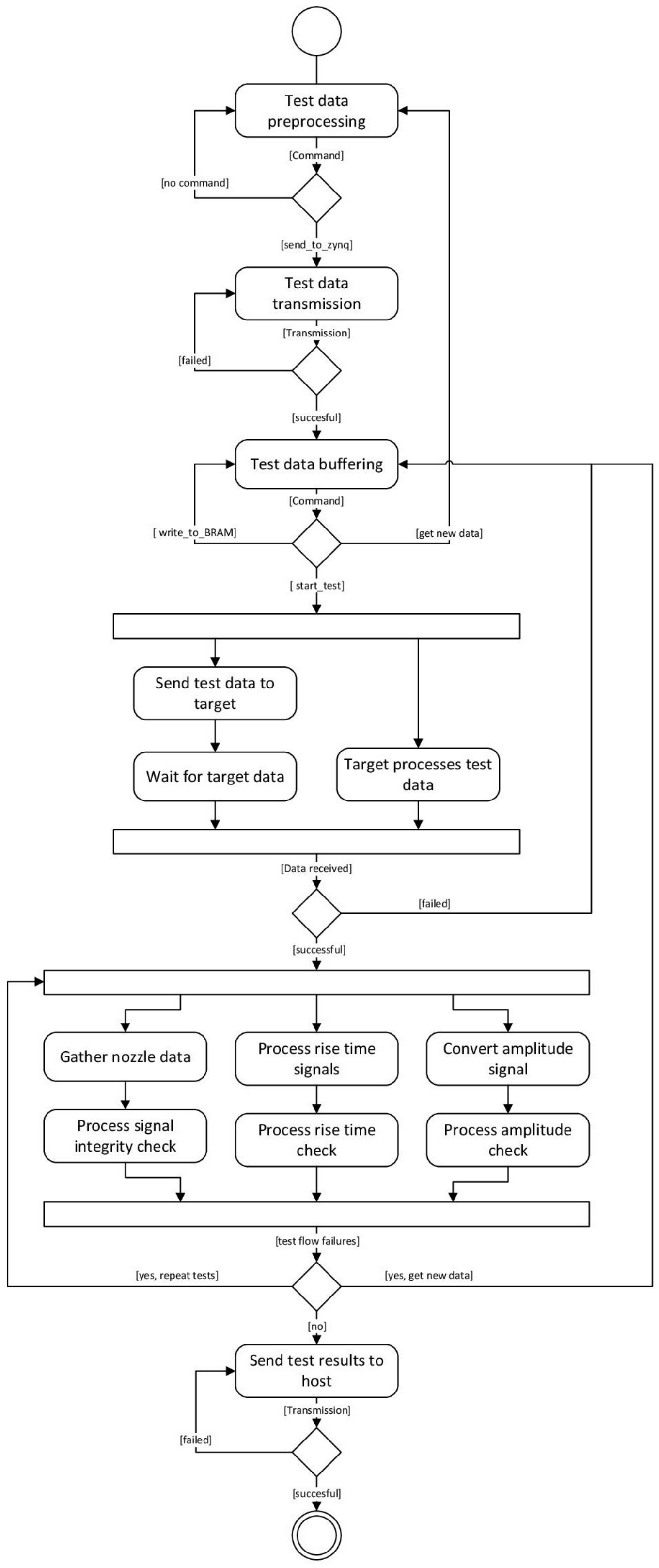


Figure 4.3: Activity diagram of the final stage test procedure

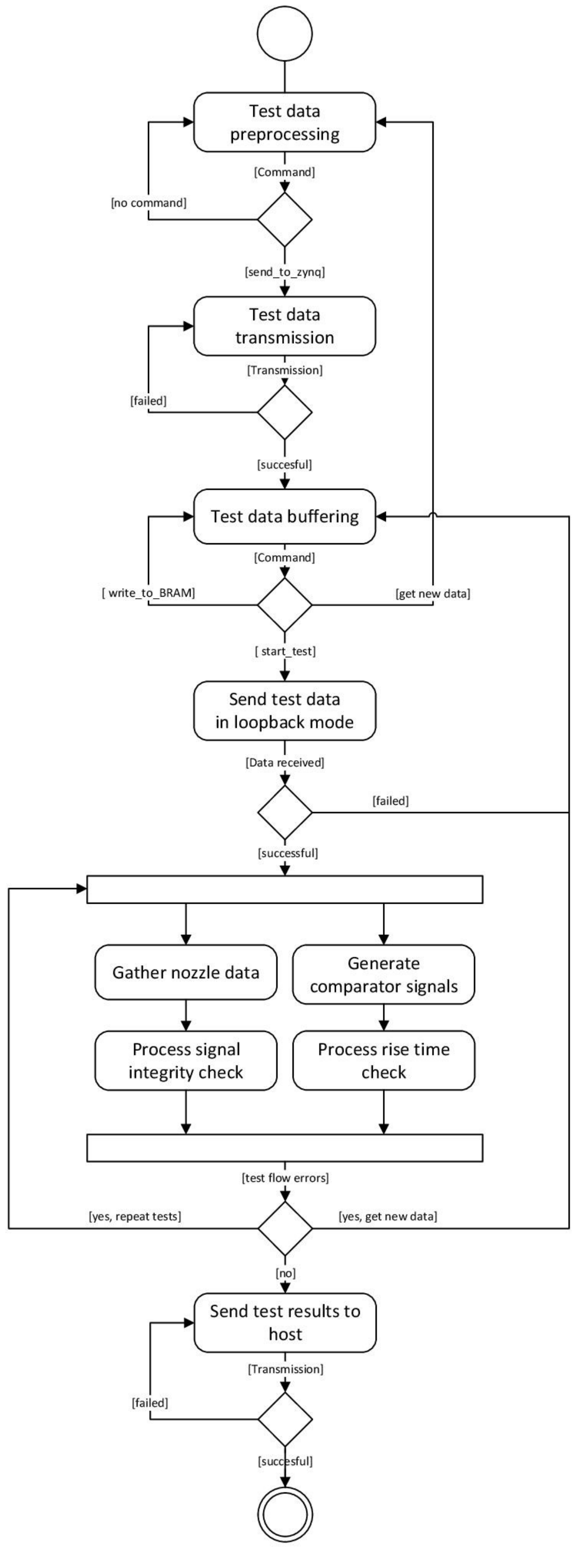


Figure 4.4: Activity diagram of the current stage test procedure

4.2.3 Interaction description

For the visualization of the test procedure module interaction, a UML sequence diagram is used. It shows basically the interaction of the communication partners (objects) via messages. Each object has a so called lifeline which illustrates the activity of the object. Horizontal arrows represent the exchanged messages.

Here also a differentiation between current status with the PicoZed platform and the final state test module PCB is necessary. Therefore the two respective sequence diagrams are shown in Figure 4.5 and 4.6. First of all the user has to handle the preprocessing of the test pattern on the host PC. After that the host requests to store the data on the test module. The control logic of the Zynq PS then communicates with the PL to write the test pattern into the specific BRAM cell. When this is finished, the test module confirms the successful storage operation.

Now again a user action is necessary to start the test procedure on the host PC. This transfers then the start command to the test module, where the PS control logic on the Zynq prompts the TDG to read out the test data from the BRAM. Now the further commands depend on the used hardware platform.

Within the final test module the TDG now sends the test pattern to the target electronics, which then processes the data. The test module waits until the target has finished and sent the signals for the print-heads. Now the according PL component executes the data line test and transfer it's results to the PS control logic. In parallel the respective PS components receive the data from the target for the amplitude and rise time tests, which are then also executed.

In the current implementation the GTP module is configured in loopback mode. Thus, when the GTP receives the transmitted test pattern, he directly forward the nozzle data to the data line test component in the PL, which processes the data line test. In parallel, the CSG starts with the pulse generation for the rise time test. When the PS recognizes this pulses by the raised GPIO interrupts, it processes with the rise time test.

When all tests are finished, the PS sends back the results to the host PC, where the user has to verify them.

4.2.4 Dataflow description

The dataflow diagram in Figure 4.7 visualizes the possible data paths through the system and how the particular data is processed. Rectangles denote entities and circles denote processes. A memory cell is build of two parallel lines. The different colors indicate the respective type of data. The grey items again highlight the external part of the test module PCB implementation. Basically the whole dataflow consist of a small set of processes:

Transfer The transfer operation mainly exchanges data between two entities without modifying the content. In the real implementation it can be either a simple signal or bus connection within the PL or a dedicated software function of the control logic.

Load After buffering the test pattern in the PS, the load operation fetches the data from the buffer memory and forwards it to the PL-interface.

Store This process is also used in the PS and PL. In both cases it saves the test data in the appropriate memory cell.

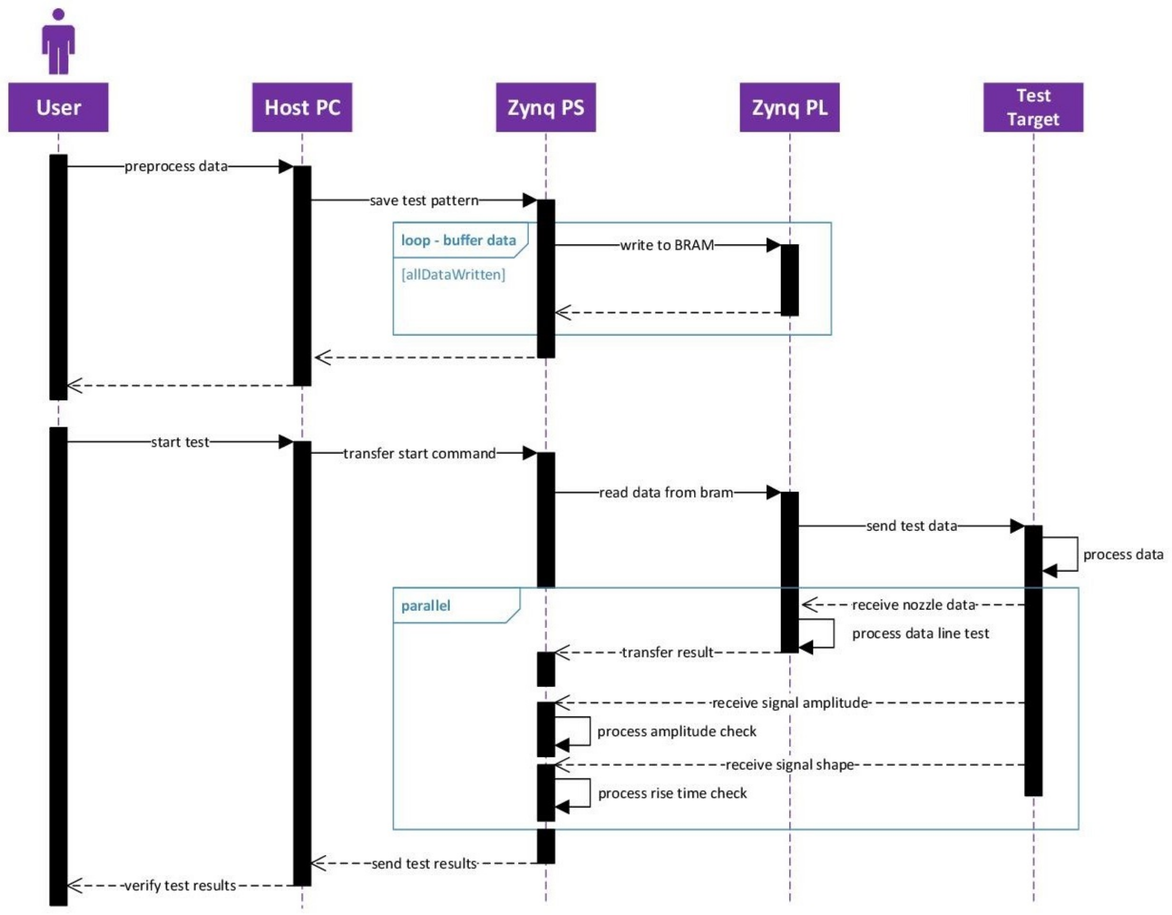


Figure 4.5: Sequence diagram of the overall test procedure

Read With a read command from the TDG this process fetches similar to Load the data from the BRAM cell in the PL.

Write It simply writes the data from the TDG into the BRAM cell.

Send With this process the data is on the one hand sent from the GTP to the test target. On the other hand without external components it is sent internally to the TDG.

Generate This operation produces the pulse signals for the comparator simulation with the CSG and is only available in the PL.

Verify The verification procedures are for the different test cases.

4.3 Host communication

The whole test procedure is initialized and started over particular commands at the host PC. The primary communication channel between the user and the test module is a UART interface, which is electrically implemented as RS485 communication standard. It serves

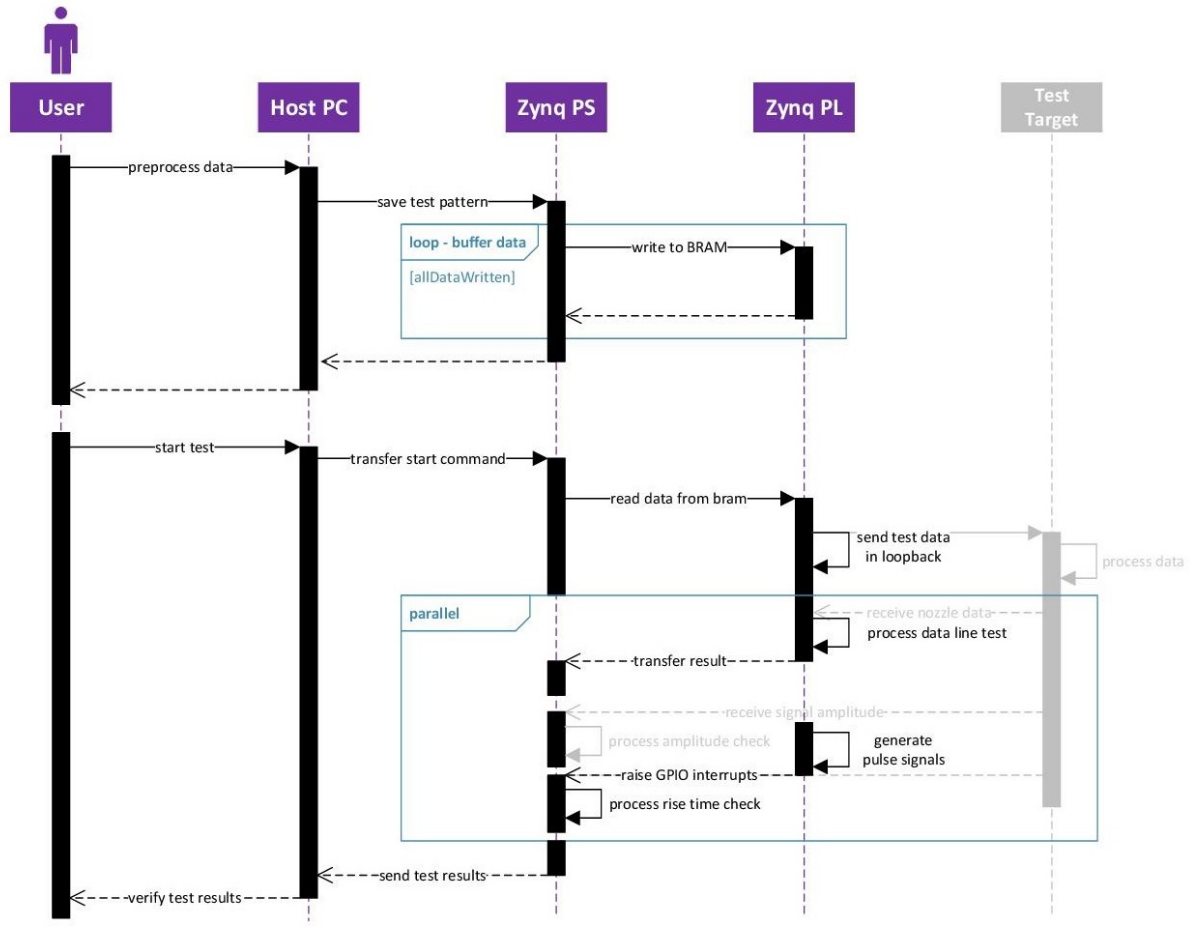


Figure 4.6: Sequence diagram of the current test procedure

mainly for transmitting the test pattern, control commands and test results. The second channel is also a UART interface at the test module site, but basically a simple serial port interface at the host site and only used for debug purposes on the final test module PCB. Since the RS485 communication is not available on the PicoZed platform, the intended purpose of these two channels has been changed. In the current implementation stage the serial port UART, connected via USB, is configured as major communication channel for both transfer and debug purposes. A JTAG connection is additionally used as development and debug interface to the Zynq HW/SW systems.

4.3.1 Serial port communication channel

On the PicoZed platform an embedded UART-to-USB converter IC[11] is used as protocol bridge. With the appropriate drivers on the host site, the communication can be established with any terminal program with serial port capabilities. The standard parameter values for the serial port channel configuration can be found in Table 4.1. Currently there

[11]https://www.silabs.com/documents/public/data-sheets/cp2104.pdf

is no dedicated user application on the host site, because all necessary commands and data can be easily transmitted via the terminal program. The only special requirement of the used application is the capability of sending text files. A such suitable terminal program with easy user interface is CoolTerm[12]. It features not only the mentioned text file sending possibility, but also a HEX viewer and multiple concurrent connection establishment as well as direct recording of received data into text files. The latter feature could be used to implement automated test result logging, which can be very useful for multi-target tests as mentioned in subsection 4.1.2.

Parameter	*PicoZed values*
Baud rate	115200
Data with	8 bit
Parity bit	none
Stop bit	1 bit
Flow control	none

Table 4.1: PicoZed serial interface configuration parameters

4.3.2 Test data transmission

The above mentioned UART communication channel is primary used for the test data transmission from the host PC. To send to and receive data from the host, simply the predefined functions of the BSP can be used. As the UART interface is also configured in interrupt mode, the particular handler routine implements the read and write control logic of the FIFO buffers. A detailed description of the necessary interrupt configuration can be found in [Xil16b].

If a UART interrupt is raised at any time, the respective handler routine first checks the type of interrupt. This could either be a successful send or receive operation, a full or empty FIFO or an occurred transmission error. According to the type, the handler can react with the appropriate subroutine. When the host starts with sending test data packets, the handler routine is called when the first 64 bytes are successfully received. This data is stored in the buffer memory, the RX-FIFO and it's interrupt flag is cleared and the next 64 bytes can be received. This is repeated as long as the whole test pattern is transmitted. The same procedure is executed the other way around when sending the test results to the host, whereas this normally needs only a few cycles. If any error occurs during transmission, the handler routine can detect it according to the interrupt type and request a send-repetition of the corrupted data packet.

[12]http://freeware.the-meiers.org/

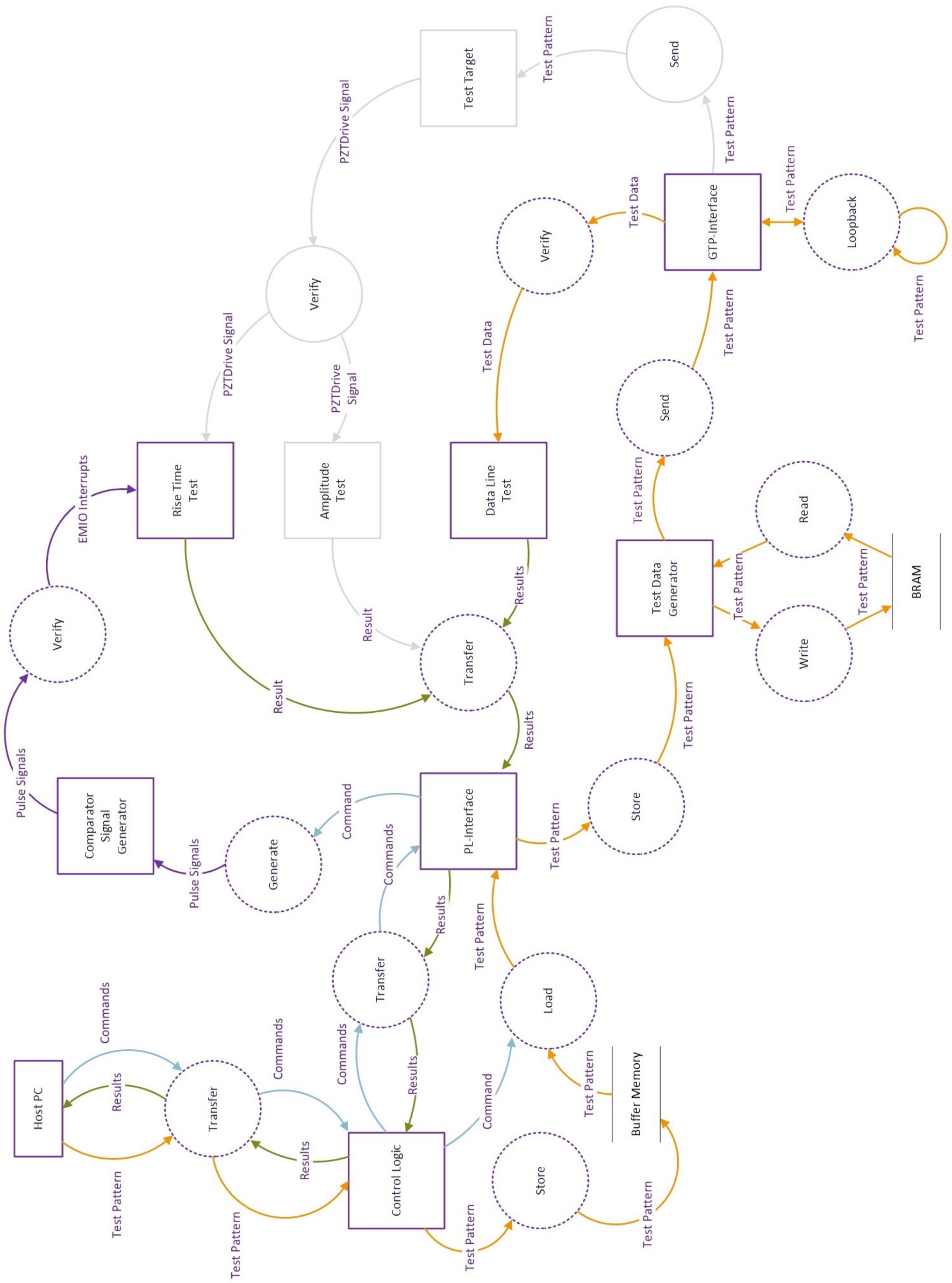


Figure 4.7: Overall dataflow diagram

4.4 Zynq processing system components

As one can see in Figure 4.2 in subsection 4.2.1 the processing system has several components, which are basically software functions for the dedicated hardware modules of the ARM PS. In the Xilinx design flow first the required hardware is implemented and synthesized, then the appropriate software functions are generated within the Xilinx SDK. All necessary software libraries are provided as so called *Board-Support-Package* (BSP). There the different initialization, configuration and communication functions for the particular hardware devices of the Zynq PS are already implemented. With the register and function definitions of the BSP, setting up and programming of the required functionality is quite straight forward. The following sections describe the software implementation of the PS components in more detail.

4.4.1 ARM firmware structure

Due to the special architecture of the Zynq SoC and the high performance processing system with a dual core ARM processor, the major tasks of the whole test procedure such as control and supervising functions as well as the test case execution can be implemented in software. As the ARM firmware is completely written in C language, several functions for the particular tasks are developed.

GPIO setup: These functions initialize the necessary PS and PL GPIO pins. During HW and SW development the PicoZed carrier card peripherals such as push buttons and LEDs were used for debug and control purposes. Therefore the direction and init-value of these GPIOs need to be preconfigured.

Interrupt system setup: The whole Zynq firmware is implemented as interrupt-driven processing system. This has the benefit, that unnecessary polling loops do not block the ARM cores and waste processing resources. Therefore the GPIO, timer and UART interface interrupts need to be configured within this setup functions.

Interrupt handers: The above mentioned peripheral interrupts need also associated handler routines. These functions mainly implement the test case executions for rise time and amplitude checks. Furthermore the UART interrupt handler controls the host communication and data buffering within the control logic.

PL-Interface: This function represents the software counterpart of the VHDL component in the PL. Here all necessary commands for controlling the dataflow within the PL are defined. The function is mainly called by the particular interrupt routines, which communicate with PL components.

4.4.2 Main control logic

With basically all these software functions the main control logic is build up. The application is placed within one C file, which includes all the required driver libraries of the BSP and the function declarations. As previously mentioned, the system is configured as interrupt driven software application on the ARM processor. In the main function basically only the initialization and setup part is executed. The rest of the control logic is embedded in the interrupt handler routines.

Initialization phase

First of all the Zynq platform is initialized with a Xilinx-specific **startup function**. Then the configuration of the used GPIOs in the PL followed by those PS is done. The latter one includes especially the configuration of the EMIOs which are routed into the CSG part of the PL for the rise time test. The next step is to **setup the interrupt system**. Here the GIC is initialized as well as several instances of GIC-specific variables and structs, which are then assigned to the respective pointers and functions. Finally the global ARM interrupts get enabled.

After the basic interrupt system setup, the specific configuration for the PS and PL **GPIO-interrupts** is done. These functions are quite similar, because the core setup is nearly the same for both parts. With the appropriate functions from the BSP the interrupt trigger is defined to rising edge sensitivity. Then the corresponding pointers to the GPIO interrupt handler addresses are assigned and followed by connecting the GPIO interrupts to the GIC. The GPIO interrupt enabling rounds off this function.

A similar but slightly more comprehensive setup procedure is necessary with the **TTC configuration**. Currently only one of three timer/counters are used. Therefore the timer 0 and it's interrupts are initialized and connected to the GIC. The TTC features several working modes which are explained later on. For the rise time test basically the standard counter mode is suitable and configured here. When using the match mode also the respective match register values have to be preloaded here. After that the timer can be reseted and started.

The final initialization step is to configure the UART interrupts. Therefore the appropriate pointers have to be initialized and the handler routine has to be connected to the interrupt subsystem. After that an interrupt-mask has to be set up with the required parameters such as FIFO-empty and FIFO-full flags for receiver and transmitter ports. Furthermore the UART mode register must be configured to normal operation. Before the initialization phase can be finished, the receiver and transmitter FIFOs have to be initialized with the appropriate start values.

Interrupt driven application phase

For controlling the test module, the intelligence has been moved to the interrupt routines. After the initialization phase, the ARM core goes into the WFI-state. As long as no interrupt occurs, the processor is not blocked and can process other tasks, but in the current implementation stage it just is running in idle mode to save power. With further development during the porting onto the test module PCB, dedicated power saving mechanisms of the Zynq could be integrated instead of the WFI instruction. When an interrupt occurs, the generic interrupt controller (GIC) looks for the respective handler routine and executes it. After successful execution the core goes back into WFI-state.

4.4.3 Interrupt logic

The Zynq SoC has a state-of-the-art interrupt control system, the GIC. It can handle several different internal and external interrupt sources. Following general interrupts are considered by the GIC [Tay17]:

Software-Generated Interrupts (SGI): The two processor cores have up to 16 software-generated interrupts per core. They can either interrupt only one of them or both together.

Shared Peripheral Interrupts (SPI): These interrupts can come from the I/O peripherals and are shared between both ARM cores. Furthermore these interrupts can be configured to come from and to PL components such as custom IP cores.

Private Peripheral Interrupts (PPI): Here five interrupts can be exclusively used by each CPU such as CPU timer, watchdog or dedicated PL-to-CPU interrupts.

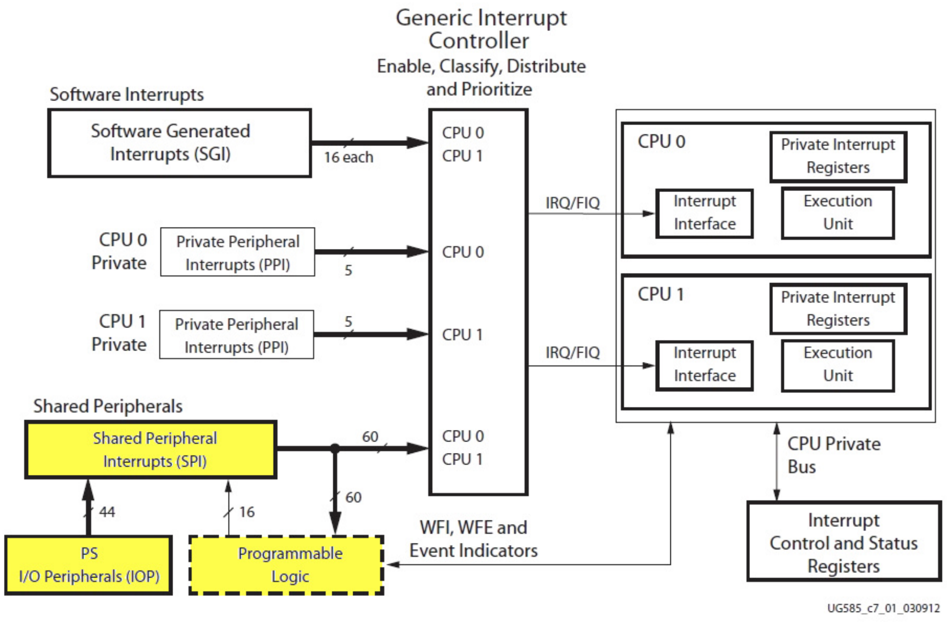

Generic Interrupt Controller
Enable, Classify, Distribute and Prioritize
Software Interrupts
Software Generated Interrupts (SGI)
16 each
CPU 0
CPU 1
CPU 0 Private
Private Peripheral Interrupts (PPI)
5
CPU 1 Private
Private Peripheral Interrupts (PPI)
5
Shared Peripherals
Shared Peripheral Interrupts (SPI)
60
60
44
16
PS I/O Peripherals (IOP)
Programmable Logic
WFI, WFE and Event Indicators
IRQ/FIQ
IRQ/FIQ
CPU 0
Private Interrupt Registers
Interrupt Interface
Execution Unit
CPU 1
Private Interrupt Registers
Interrupt Interface
Execution Unit
CPU Private Bus
Interrupt Control and Status Registers
UG585_c7_01_030912

Figure 4.8: Block diagram of the interrupt logic on system-level [Xil16b]

The current implementation uses only the **shared peripheral interrupt** source as also highlighted in Figure 4.8. The most important sources for this system are the UART interrupts for host communication and the internal EMIO interrupts.

The UART interrupts are raised every time when a new command or data set from the host is received. Then the appropriate command is handled by the interrupt routine, which continues with executing the respective program part. If a new data set is received, the control logic temporarily stores this data into the buffer memory until further commands are received.

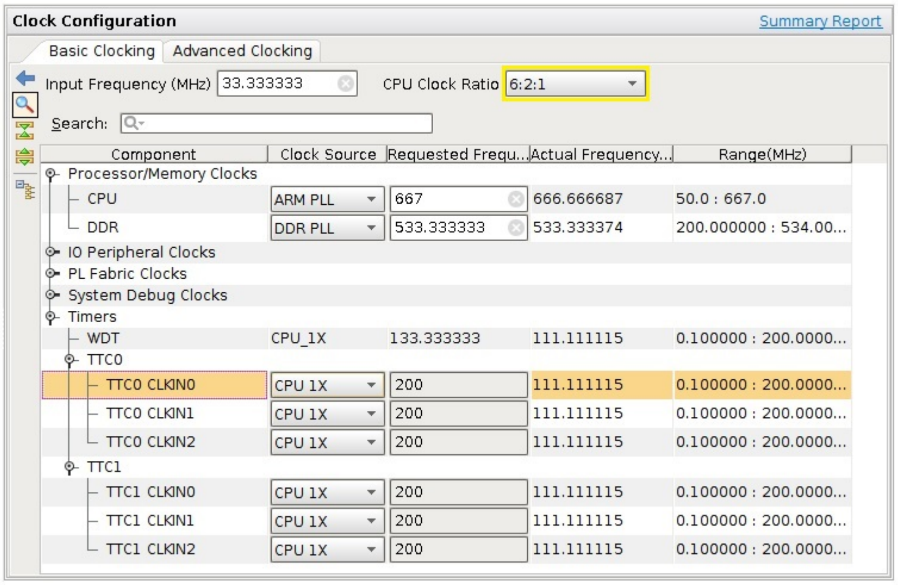
The EMIOs used for the rise time test are directly routed to the CSG component in the PL. With appropriately configuring the necessary EMIO registers, the CSG output signals can trigger the specific EMIO interrupts at rising edges. With this setup, a future porting to real GPIO pins on the test module PCB would only require a modification in the GPIO interrupt configuration.

Currently the timer interrupts are only used for debug purposes and other miscellaneous tasks such as executing non-blocking wait statements or test procedure duration measurement. For the rise time calculation no timer interrupts are necessary at the moment. This is covered in more detail in the following section.

Also the standard GPIO interrupts are used for less important subtasks like basic user interaction with the push buttons and status indication with LEDs directly on the PicoZed platform.

4.4.4 Triple Timer Counter

Beside 32 bit private timers for each core and one shared 64 bit global timer, the Zynq PS has also two TTC instances. Each TTC features three independent 16 bit timer/counters with their own 16 bit prescalers. The clock input for each of the 6 timer/counters is also independently selectable from internal PS bus clock, internal clock from PL or external clock sources from MIO. Figure 4.9 shows the clock configuration within the Xilinx Vivado Design Suite. One can see that the clock frequency of all six TTCs is set to *CPU_1X* and the CPU clock ration to *6:2:1*, which means that the PS internal prescaling ratio is set so $1/6^{th}$ for the TTCs. This results in a clock frequency for all TTC instances of about 111 MHz, which corresponds to counting steps of approximately 9 ns as already calculated in Equation 3.6. The TTC counter modules have two different modes of operation, the



Input Frequency (MHz) 33.333333 CPU Clock Ratio 6:2:1

Component	Clock Source	Requested Frequ...	Actual Frequency...	Range(MHz)
Processor/Memory Clocks				
CPU	ARM PLL	667	666.666687	50.0 : 667.0
DDR	DDR PLL	533.333333	533.333374	200.000000 : 534.00...
IO Peripheral Clocks				
PL Fabric Clocks				
System Debug Clocks				
Timers				
WDT	CPU_1X	133.333333	111.111115	0.100000 : 200.0000...
TTC0				
TTC0 CLKIN0	CPU 1X	200	111.111115	0.100000 : 200.0000...
TTC0 CLKIN1	CPU 1X	200	111.111115	0.100000 : 200.0000...
TTC0 CLKIN2	CPU 1X	200	111.111115	0.100000 : 200.0000...
TTC1				
TTC1 CLKIN0	CPU 1X	200	111.111115	0.100000 : 200.0000...
TTC1 CLKIN1	CPU 1X	200	111.111115	0.100000 : 200.0000...
TTC1 CLKIN2	CPU 1X	200	111.111115	0.100000 : 200.0000...

Figure 4.9: Zynq PS clock configuration of Xilinx Vivado Design Suite

interval mode and the overflow mode. The latter one is used in the current PicoZed setup for the rise time test. Here, the particular counter increments continuously between *0* and *0xFFFF*. The counter value register can be read out at any time, which is used for the rise time calculation as later explained in subsection 4.6.2. Each timer/counter features additionally two independent match registers. These can be configured to raise counter match interrupts, which are currently used for interrupt driven wait statements. These are applied for simple non-blocking wait functions, which are sometimes helpful to resolve timing issues.

4.4.5 PL-interface SW function

As already described, the PL-interface is the primary communication channel between the PS and PL. Therefore the respective functional counterpart is required for both sites of the channel. On the PS site the interface is basically a software function. The communication is based on read and write operations to predefined 32 bit registers, which are also accessible from the respective PL component. For accessing the register within the PS, the Zynq BSP provides the respective library functions for 32 bit read and write operations.

The whole communication channel is currently build up with three registers, two for data transmission and one for the commands. In terms of firmware development this means nothing else then writing to different addresses. Therefore each register has a predefined address offset, which can be used by the register access functions.

For the control logic of the communication channel, both partners have to know the meaning of each bit in the command register. Therefore a generic protocol header is used as seen in Figure 4.10. The higher 16 bits of the header are assigned to the command

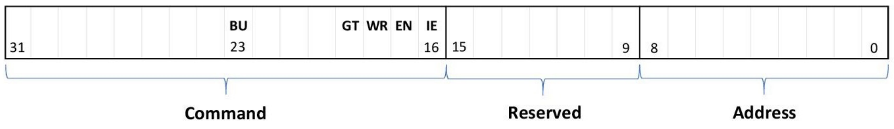


Figure 4.10: PL-interface protocol command header

followed by 7 reserved bits and the 9 address bits. In the command section the particular bits denote specific operations. Currently 5 different bits are used in certain combinations to form the necessary commands. In Table 4.2 the meanings of these bits are explained and in Table 4.3 the actual bit combinations with the respective commands are declared.

Bit abbreviation	*Bit index*	*Description*
IE	0	IDLE bit only used for returning to the idle state
EN	1	ENABLE indicator for general BRAM operations
WR	2	WRITE operation indication bit
GT	3	GTP communication with test target or in loopback
BU	7	BURST operation indicator for read/write commands

Table 4.2: Description of the command bit abbreviations

Since the channel communication uses a BRAM instance at the PL site for data buffering, the commands are respectively designed for BRAM interface compatibility. This means that the particular bits implicitly represent the corresponding BRAM control signals. Thus, e.g. for the READ command, only the enable bit is necessary, which simply activates the BRAM's output stage for read operations. The GT bit is set when high-speed GTP transmissions should be executed, such as sending the test pattern in loopback mode or to the test target. Especially in this case mostly a combination with the BU bit is used to sent a specified amount or the whole BRAM content over the GTP in burst mode. The reason why the BU bit is defined at command index number 7 and not directly after the GT bit is to have some free bits available for additional functionality.

	Binary Representation [Bit Index]					
Commmand	BU [7]	GT [3]	WR [2]	EN [1]	IE [0]	*HEX-Representation*
IDLE	0	0	0	0	1	0x01
READ	0	0	0	1	0	0x02
BURST READ	1	0	0	1	0	0x82
WRITE	0	0	1	1	0	0x06
BURST WRITE	1	0	1	1	0	0x86
GTP SEND	0	1	1	1	0	0x0E
GTP BURST SEND	1	1	1	1	0	0x8E

Table 4.3: Bit combinations for the currently used PL-interface commands

As also visible in Figure 4.10 there are several reserved bits in the header declaration. This is because they are simply not used at the moment. The lower 9 bits are used to address the BRAM data cells, since the block memory is configured to a size of 512x32 bits. If prospectively a bigger block memory is necessary, the reserved bits can be assigned to extend the possible address range.

The other two 32 bit registers for data transmission are split into a transmitter and a receiver channel with respect to the PS site. Since in the ARM firmware only sequential execution is possible, the PL-interface on the hardware site has a certain synchronization mechanism for command and data transmissions. To perform for example a write operation, first the command has to be sent over the PL-interface. Within the PL the command bits are buffered and the respective signals are set. After that the TDG listens on the data channel for a new 32 bit message and stores it at the appropriate BRAM address.

For BURST operations the data channel is opened as long as new messages are sent. Here also an initial command message is necessary to prepare the fabric signal assignment. Then the channel can be used as long new data is available. The BURST WRITE operation is indicated with the appropriate message over the command channel. Then the required amount of data messages is sent in a continuous loop of single WRITE operations within the PS function. The BURST READ is done quite similar the other way around over the receiver channel.

As previously mentioned, the operation principle of the GTP BURST SEND is a little bit different. Here the burst bit doesn't indicate a continuous bit-stream over the data channel, but rather sending a specified amount of BRAM data over the GTP interface. Thus, the data channel could still be used for other transmissions.The address bits of the command header define in this case the highest BRAM address until the GTP should read out the data.

4.5 Zynq programmable logic components

The PL part of the Zynq platform is used to implement the VHDL hardware components for the test module. In the lower section of Figure 4.2 the respective parts and their interconnections are illustrated. In contrast to the PS, where the components are basically C functions and library calls, all PL components are either custom or prebuilt IP cores or handwritten VHDL components. By the conventional VHDL design methodology all components are organized in a hierarchical structure which is shown in Figure 4.11. The

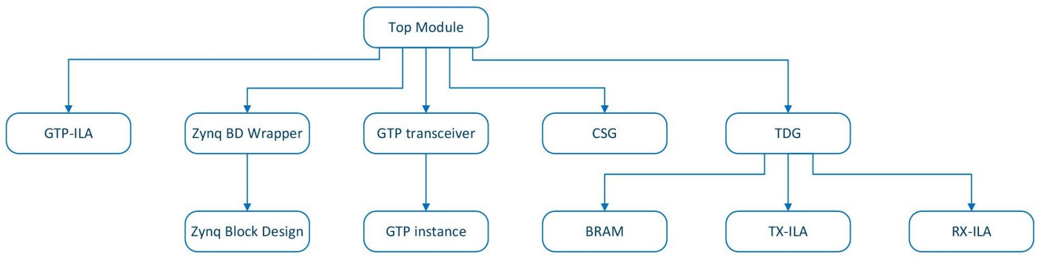


Figure 4.11: Hierarchical structure of the PL components

main components like the Zynq base design wrapper, the test data and comparator signal generators and the GTP transceiver are instantiated in the top module. Here also all interconnections between the components are made. Additionally an ILA core is placed in the top module for debugging and verification of the GTP communication, which will be described in more detail in section 4.6.1. The PL-interface HW module is integrated in the Zynq block design, since it is directly connected to the on-board AXI-bus.

Beside the control logic for the test data generation, the TDG contains also the BRAM instance, which is used for test data storage. Two further ILA cores are used here to verify the correct TDG data transmission for the RX and TX channel.

4.5.1 Zynq block design

To be able to develop working a system-on-chip design based on the Zynq platform, a basic block design is necessary. With the IP Integrator of the Xilinx Vivado Design Suite this can be achieved quite easily. The major component for this subsystem is the Zynq processing system IP core, which can be found in the Vivado IP catalog. By placing this core in the block design, the IDE automatically builds up the required base design consisting of the ZYNQ7 Processing System and a Processor System Reset block as highlighted in Figure 4.12. Also the basic signal and bus connections of the base design are wired up automatically by the block-automation tool. By double-clicking on the ZYNQ7 PS core, the entire configuration for the ARM cores, peripherals, system memory, interrupts and clocking can be done within one graphical user interface. Depending on the chosen configuration, the IP cores and interconnects of the base design are automatically adapted to necessary ports and buses.

With just the Zynq base design and no other VHDL components defined, the base design wrapper will act as top module and only the processing system will be initialized in hardware. In this case the Zynq could already be used as a conventional microcontroller without FPGA part. To gain also the advantage of the programmable logic, other IP cores and VHDL components need to be added and instantiated in a separate top module. Then the Zynq block design must be instantiated in this overall top module. Furthermore all required signals and ports need to be declared and wired up in the block design and also in the top module.

For the test module block design several custom IP cores are used. When placing them into the previously mentioned base design, the block-automation tool inserts also

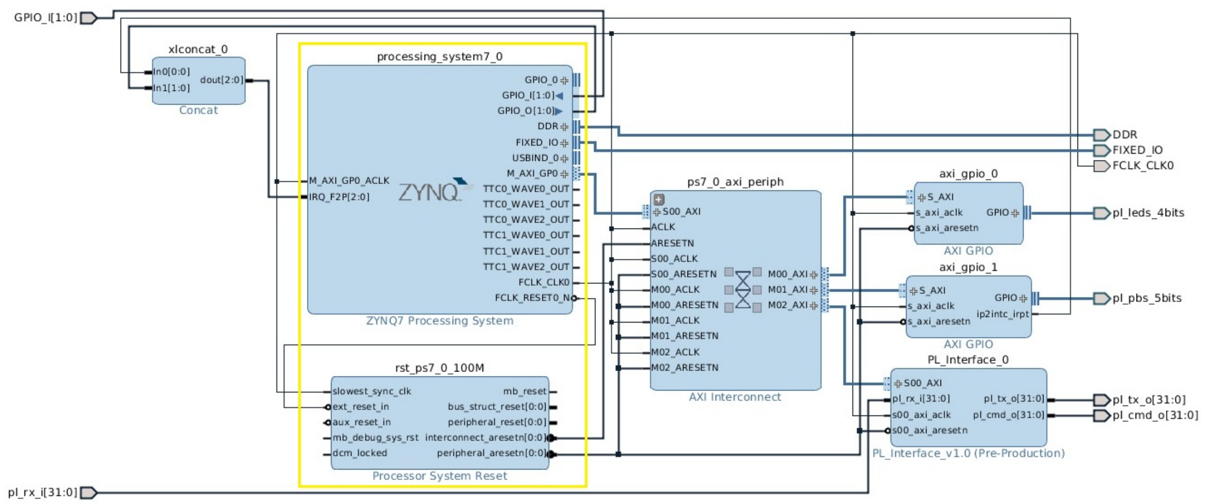


Figure 4.12: Zynq block design inside the Xilinx Vivado IP Integrator

the AXI Interconnect core and wires up the bus interface and clock signals. For accessing the buttons and LEDs of the PicoZed carrier card, which are physically connected to PL IOs, two custom cores with AXI interface have been developed. The IO access for these components is established by read and write operations to certain register addresses, similar to the PS-PL communication.

Another important component to mention is the Concat block, which acts like a multiplexer for the GPIO interrupts. Generally within the Zynq PS it is possible to configure several different sources from the PL. As described in subsection 4.4.3 the EMIO interrupt trigger can be raised by PL components. Therefore this trigger signal needs to be connected to the PS interrupt controller. One possibility to access fabric interrupt sources is to connect the PL-component trigger signal with the F2P-input port of the ZYNQ7 processing system. Since the PL-buttons generate interrupts which also need to be connected to the interrupt controller, the Concat block is necessary to map the respective signals to the PS.

4.5.2 PL-interface HW module

The PL-interface component is internally build up in the same way as the IO cores, but with three dedicated channels, RX-data, TX-data and command. As described in subsection 4.4.5 the PL-interface software counterpart implements the generic protocol header and the necessary commands. These commands need to be processed by the appropriate HW component. Therefore the PL-interface HW module implements the physical connection between the ZYNQ7 Processing System and the rest of the programmable logic by forwarding the data and commands from the PS counterpart. More precisely, the component acts like a AXI-to-register converter for all channels. The TDG's *pl_tx* and *pl_cmd* 32 bit ports are directly connected to the equivalent PL-interface ports. All further transmission logic is handled by the internal state-machine of the TDG component.

4.5.3 Test data generation

When the PS control logic sends a command over the PL-interface, the TDG component gathers the command and data bytes from the respective registers. The internal state-machine's next state logic then decides on the further data processing depending on the transmitted command header as defined in subsection 4.4.5. In Figure 4.13 the state diagram of the internal state-machine is shown. After initialization, the TDG goes into

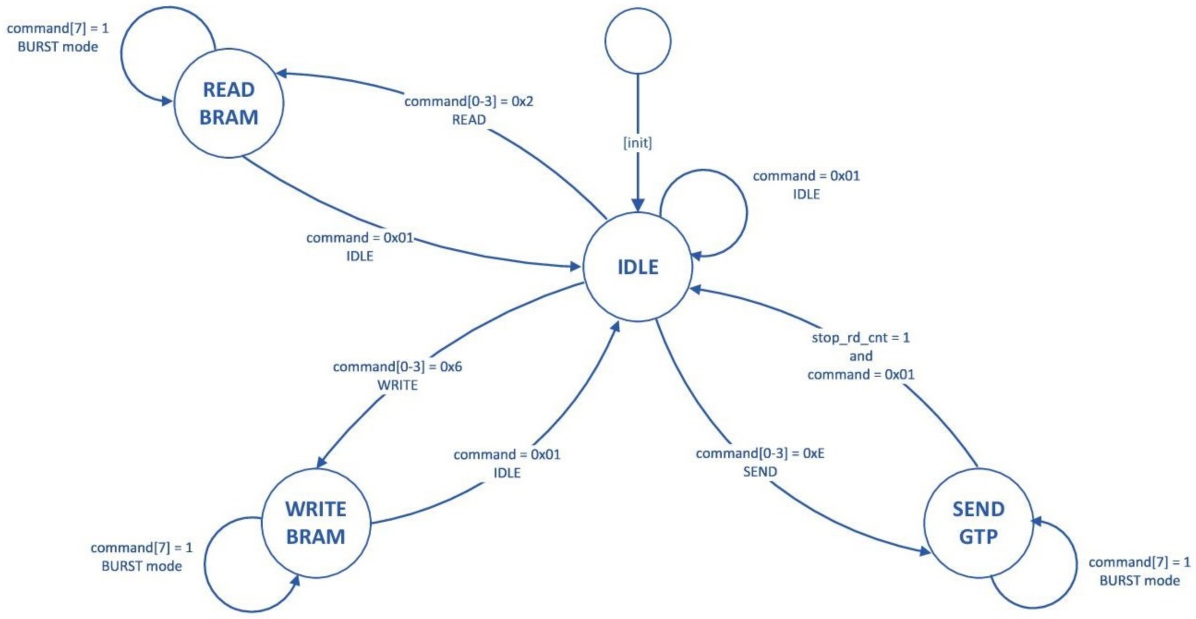


Figure 4.13: Internal state-machine of the TDG component

IDLE state until a new command is received from the PS. If that happens, the TDG first looks at the least four bits of the command header, which define the command type (see Table 4.3). Then the next state logic switches into the appropriate state. After one processed command cycle, the next state logic checks the BU-bit. If this is set, the command is intended for burst operation and the TDG continues in the current state until the particular termination criterion is reached.

In the most cases (READ/WRITE BRAM) the termination is achieved by an IDLE command. This is because the appropriate SW function executes single operations in a loop as long as data is available to be processed (see also subsection 4.4.5). After finishing the loop, an IDLE command is sent to terminate the burst operation.

To terminate the SEND GTP state in BURST mode, another criterion must be fulfilled. Therefore a closer look to the working principle of this state is necessary. As described in Table 4.3 the GTP BURST SEND command from the PS does not continuously occupy the data channel, but rather read a certain amount of BRAM data to send over the GTP. Within this command the address field denotes the highest BRAM read address. The TDG uses this number as counter value, which is continuously decrementing during the GTP SEND operation. When the read counter reaches zero, the *stop_rd_cnt* flag is set to *1* to indicated a completed BRAM-to-GTP transfer. Thus, this flag can be used to terminate the SEND GTP state in BURST mode. Additionally an IDLE command is necessary, because otherwise the next state logic would repeat the GTP data transfer.

4.5.4 Test target communication

As already discussed is the final goal of the data acquisition and test platform to perform certain test cases with the print-head control electronics. Therefore a proper communication channel is needed. The Zynq platform features four serial gigabit transceivers with up to 6,25 Gbit/s. Each GTP can be configured within the Xilinx 7 FPGAs Transceiver Wizard[13] for several different industry standard protocols such as Aurora 8B/10B, DisplayPort, Serial RapidIO or PCI Express on certain devices.

GTP interface

Since the print-head control electronic implements a standard 8B/10B enconding scheme for data exchange, one of the four GTP channels of the PicoZed platform is also configured for this protocol. In the Vivado Transceiver Wizard tool the major parameters for the generated IP core like reference clock source, GTP channel, communication protocol, character alignment etc. can be predefined. For the current implementation the GTP instance is configured as seen in Table 4.4. It is instantiated and wired up in the top

Parameter	*Value*
Protocol	Standard 8B/10B
Line Rate	2.5 Gbit/s
Reference clock	125 MHz
Data Width	16 bit

Table 4.4: Configured parameters of the used GTP instance

module. As clock source for the internal logic of the configured core, a specially generated 100 MHz user clock of the Zynq PS is routed to the PL and connected appropriately. This source is synchronized with the AXI-bus-clock of the Zynq block design and drives also the rest of the PL components. The GTP core generates automatically two user clocks for the transmitter and receiver, in this case at 125 MHz. This results in two asynchronous running clock domains within the PL. This is generally no problem, but since the GTP TX-data input port must be fed synchronous with the TX user clock, it must be considered. Therefore the BRAM instance is configured with a dual port interface with independent clock sources. The port A is clocked by the PS user clock and the port B by the TX user clock. With this mechanism the GTP transmitter input can be fed every TX user clock cycle with 16 bits of data to send to the target.

8B/10B encoding

Due to the very high data rates of the GTP interface, a 8B/10B encoding scheme[14] is used for test data transmission. The 8B/10B code is a communication standard for high-speed

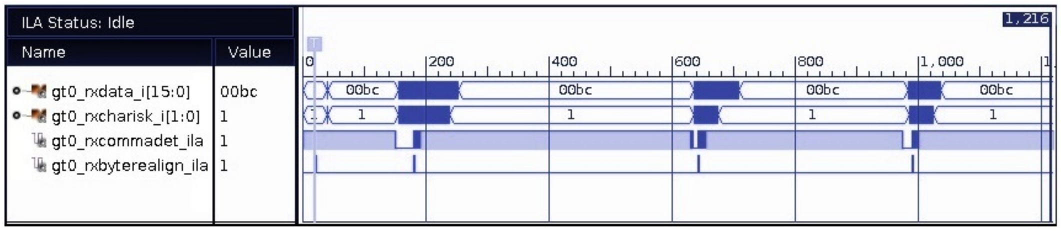
[13]https://www.xilinx.com/support/documentation/ip_documentation/gtwizard/v3_3/pg168-gtwizard.pdf

[14]http://www.itwissen.info/8B-10B-Codierung-8-binary-10-binary-8B-10B.html

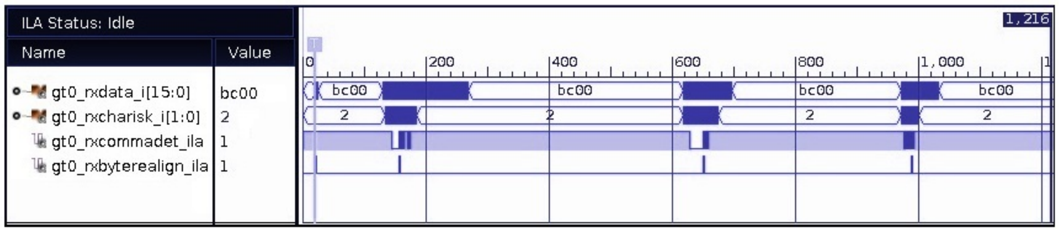
transmission lines, which encodes 8-bit-data with 10 bit symbols. By knowing that 10 bit can represent 1024 values, 8 bit just 256 and considering only the 10-bit combinations with maximum 5 ones (or zeros) in a row, a signal stream with a high number of level changes can be achieved. This results in a well DC-balanced transmission line and enables clock recovery at the receiver site. Furthermore when subtracting the necessary 10-bit combinations for the 8-bit data, the remaining 768 combinations can be used for special character and control symbols. The latter ones are called K-Characters and can be found in Table B.1 in Appendix B.

Comma-alignment

A very important symbol within the K-Character set is the comma, which is usually used for synchronization of the data stream. This means that a comma symbol at some point of the stream indicates the beginning of a data word. The receiver can detect the comma symbol and automatically align the payload words withing the stream. In the current implementation when no payload is available, the comma character *K.28.5* is send continuously over the GTP interface. Thus, the receiver can synchronize its clock to the stream and align new payload data faster if available. When configuring the GTP core with



(a) Comma at lower data byte



(b) Comma at higher data byte

Figure 4.14: ILA-visualization of the comma alignment principle

the Transceiver Wizard, the valid characters for the comma can be predefined. For some applications it might be useful to have multiple valid comma values, but in this case only the *K.28.5* is accepted. In Figure 4.14 the comma detection and data stream alignment of a test pattern transmission on the PicoZed platform is illustrated. The signals have been captured with an ILA core, which is described in section 4.6.1. The two captures differ from each other by the comma-position in the data word, indicated by the *rxcharisk.* signal. Furthermore the two signals *rxcommadet* and *rxbyterealign* indicate the detection of a comma and resulting realignment of the data stream. The *rxdata* HEX-value *0x00bc* denotes *00111101* in binary representation, which is the *K.28.5* character (referred to

Table B.1). When the project is progressed to the test module PCB porting phase, also other K-Characters will be used in the test target communication for configuration and control commands.

4.6 Test cases

In the last section a short insight into the most important test cases is given. As already discussed are signal data and shape verification the main use cases of the final test module PCB. In the current implementation a scaled down version of them is implemented and tested.

4.6.1 Data signal verification

The primary test case of the module is to verify the correct signal transmission over 128 data lines from the test target. Normally these signals include continuous enable/disable information for each of the 2048 print-head nozzles. For the signal integrity check is not the whole amount of nozzle data necessary, but at least several transmission cycles for each data line. Therefore this test case should include toggling single data lines with predefined test patterns on the final test module. The print-head control electronics is then configured in a transparent-mode, so that the predefined test pattern is sent to the target, processed there and sent back over the data lines in a known scheme.

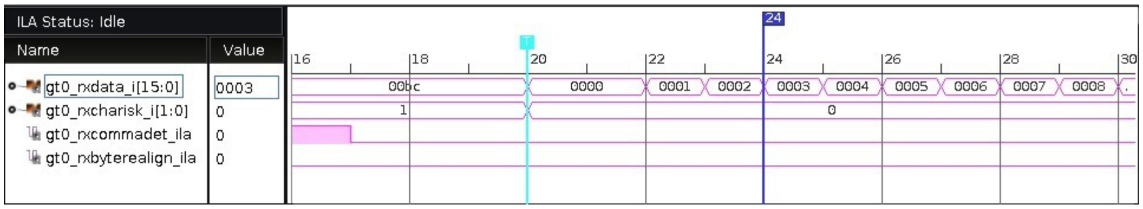
For the implementation on the PicoZed platform the test pattern is also equally received as it is sent. Therefore the later on adaption should be quite straight forward. The difference in the current setup is that the GTP receiver channel gathers the loopback data and forwards it to the data-line test instead of direct data acquisition there via the IOs. Furthermore the signal test procedure itself is different in the current development stage. Instead of working with a dedicated data signal test component, the integrated logic analyzer functionality of the programmable logic is used.

ILA cores

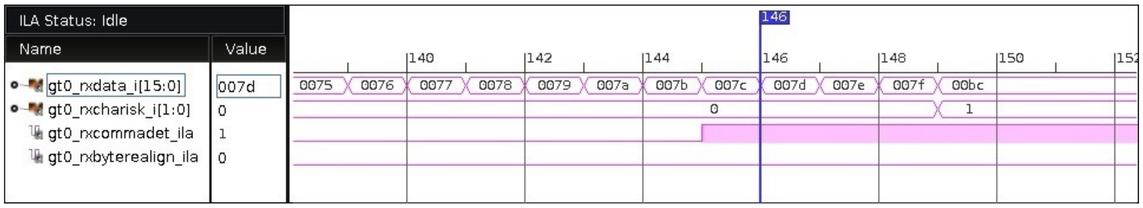
The IP Catalog withing the Xilinx Vivado Design Suite offers customizable ILA cores with up to 4096 probes per core [Xil14]. Each ILA instance can be clocked independently, but it must have the same clock source than the examined signals. For the data line test the particular ILA core is clocked by the GTP-generated RX user clock.

The data signals are buffered first in the TDG-BRAM. When the test is started throughout the system, the TDG reads out line per line from the BRAM and splits the 32 bit data into 16 bit packets, which are then sent by the GTP in loopback mode. The ILA core is connected to the receiver output signals *rxdata*, *rxcharisk*, *rxcommadet* and *rxbyterealign*. Thus, the correct data transmission can be analyzed with the logic analyzer tool of the Hardware Manager within Vivado.

In Figure 4.15 the result of the data line test can be seen. For this test case a counter in the PS firmware generates increasing values from 0 to 127 (in hex values from 0x0001 to 0x007f) and writes them into the BRAM. After that, a BURST SEND GTP command is executed, which sends the 128 data packets in loopback mode. With the receiver-ILA core the test pattern can be finally verified. In both captures it can be clearly seen, that



(a) Start of the receiving data stream



(b) End of the receiving data stream

Figure 4.15: Verification of the received data for the data line test

the *rxcommadet* signal switches its level several cycles before the transition happens. This is a characteristic of the GTP transceiver core, whereas certain output registers delay the effective signal appearance. Another pint to mention is the continuous reception of comma characters before and after the payload transmission.

4.6.2 Rise time test

Another important test case is the rise time check of the PZTDrive fire-pulse voltage, which normally drives the drop ejection in the print-head nozzles. The PZTDrive signal normally has a potential of -36 V with respect to PZTCommon on the print-head. To be able to analyze this signal with the test module PCB, a specific OPA and comparator circuit was developed (see subsection 3.4.3). When the PZTDrive signal edge passes two predefined threshold voltages, the comparator circuit's output signals switch to high at the respective points in time. The time difference of these signals can be measured within the Zynq PS and further on the rise time can be calculated.

To be able to implement the core functionality of this test case without certain hardware circuits, the CSG component generates similar signal edges within the PL. With proper internal connections of these signals to the respective PS GPIOs, the same functionality can be simulated. Therefore the CSG is continuously counting up to an interval of 2 μs. At 50 ns and 700 ns a dedicated signal switches to HIGH level, which raises GPIO interrupts. The appropriate handler routing reads the current TTC counter value register and calculates the time difference of the GPIO interrupts. This value can then be compared with the predefined signal time difference.

4.6.3 Amplitude test

The third major test case is the PZTDrive amplitude check. Unfortunately this test is not realizable on the PicoZed platform. Nevertheless a few words can be said to explain the intended functionality. With the above mentioned OPA and comparator circuit the

PZTDrive signal first is inverted and scaled down to a range between 0 V and 3,3 V. For the amplitude check this signal is then fed into a high-speed ADC. Hence the test module has to measure up to eight such signals in parallel, also the same amount of ADCs is necessary. The conversion results have to be transmitted to the Zynq PS, which is realized by a multi-slave SPI interface.

SPI chip-select encoding

In most cases of using the SPI protocol as sensor interface, only a single-master-single-slave approach is used. Since the Zynq PS has to handle eight slaves a proper chip-select logic is necessary. Fortunately the SPI interface of the Zynq PS is featuring three dedicated CS-outputs, which can be configured to encode up to eight different slaves. Additionally a 3-to-8 encoder IC is used on the test module PCB to provide the required signals for all ADCs.

The data acquisition of the converted data is then executed chip-by-chip. With switching through the respective CS-signals, the results of all eight ADC-conversions can be gathered serially over the MISO line.

Chapter 5

Concluding words

In this master thesis I have proposed and developed a state-of-the-art processing platform for high-speed data acquisition systems, which can be used as test module for the Samba print-head control electronics of digital printing solutions. With the evaluation of a well-known and practically proven platform, which embeds a Xilinx Artix-7 series FPGA with a soft-block MicroBlaze processor, and further on comparison with a Xilinx Zynq SoM with dedicated hard-block ARM cores, it could be shown, that the processing power highly influences the HW/SW separation of functional parts. Due to the fact that the specifications didn't require an algorithm-intensive solution but rather a high dataflow based implementation, a lot of the data-processing tasks could be moved into the firmware part of the processing platform. Hence the HW development phase could be limited to basic components such as the GTP transceiver and test data generation logic, which saves a lot of time of struggling with VHDL code. As a consequence the development of the remaining components in C could enhance the overall prototype process due to a high amount of predefined and prebuilt SW libraries for the particular HW functionalities of the Zynq processing system.

In the current implementation stage of the overall test module, the core functionalities could have been implemented on a PicoZed system on module with respective carrier card. This platform proved well for rapid prototyping and functional verification and is generally intended for low time-to-market development strategies due to direct system integration capabilities. Thus, the porting to the final test module PCB should be quite straight forward. This PCB was developed during a previous seminar work and followed up within this thesis. With a Zynq Z-7015 SoC as main processing platform and special electronic units like ADCs, OPAs and comparator circuits, the test module PCB is ready for integration into the targeted printing system environment. As the PicoZed platform doesn't feature some of this specific components for covering all the test cases, the core functionality was proven in a simplified way with the main data acquisition use case, the data line check. Therefore the GTP transmission was implemented in loopback mode, to eliminate the need of external test HW. The corresponding data verification was successfully done with Xilinx ILA cores within the Vivado Design Suite. For another test case, which calculates the PZTDrive rise time, the respective interrupts were triggered internally by a custom VHDL component. With using the Zynq EMIO functionality the test case could be successfully simulated by using similar core functionality.

Appendix A

Terminology

A.1 Definitions

ASIC	Application Specific Integrated Circuit
AXI	Advanced eXtensible Interface Bus
BRAM	Block Random Access Memory
BSP	Board Support Package
CMYK	Cyan Magenta Yellow Key color space
CS	Chip Select
CSG	Comparator Signal Generator
DDR	Double Data Rate
DIP	Dual Inline Package
DPI	Dots Per Inch
EEPROM	Electrically Erasable Programmable Read Only Memory
EMIO	Extended Multiplexed Input/Output
F2P	Fabric to Processign system
FET	Field Effect Transistor
FIFO	First In First Out
FOC	Fiber-Optic Cable
FPGA	Field Programmable Gate Array
FSBL	First Stage Boot Loader
GIC	Generic Interrupt Controller
HW	Hardware
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
ILA	Integrated Logic Analyzer
IP	Intellectual Property
JTAG	Joint Test Action Group
LDO	Low Drop Out voltage regulator
LSI	Large Scale Integration
MGT	Multi Gigabit Transceiver
MIO	Multiplexed Input/Output

MSPS	Mega Samples Per Second
OPA	Operation Amplifier
OS	Operating System
PCB	Printed Circuit Board
PCIe	Peripheral Component Interconnect express
PHY	Physical Layer, specialized integrated circuit
PL	Programmable Logic
Pmod	Peripheral module
PoC	Proof-of-Concept
PS	Processing System
PZT	Piezoelectric ceramic of lead Zirconate Titanate
QSPI	Quad Serial Peripheral Interface
RTOS	Real Time Operating System
SD	Secure Digital memory card
SFP	Small Form-factor Pluggable
SoC	System on Chip
SPI	Serial Peripheral Interface
SW	Software
TDG	Test Data Generator
TTC	Triple Timer Counter
UART	Universal Asynchronous Receiver Transmitter
UML	Unified Modeling Language
USB OTG	Universal Serial Bus On-The-Go
VHDL	Very high-speed integrated circuit Hardware Description Language
WFI	Wait For Interrupt

Appendix B

8B/10B Valid K-Characters

Special Code Name	*8-bit-format*	*10-bit-format*	*Alternative*
K.28.0	00111 000	001111 0100	110000 1011
K.28.1	00111 100	001111 1001	110000 0110
K.28.2	00111 010	001111 0101	110000 1010
K.28.3	00111 110	001111 0011	110000 1100
K.28.4	00111 001	001111 0010	110000 1101
K.28.5	00111 101	001111 1010	110000 0101
K.28.6	00111 011	001111 0110	110000 1001
K.28.7	00111 111	001111 1000	110000 0111
K.23.7	11101 111	111010 1000	000101 0111
K.27.7	11011 111	110110 1000	001001 0111
K.28.0	10111 111	101110 1000	010001 0111
K.28.0	01111 111	011110 1000	100001 0111

Table B.1: 8B/10B Control K-Characters

Bibliography

[Avn17] Avnet. *Design it or Buy it? Avnet's ready-made SoC modules can shorten your development cycle.* Avnet, 2017.

[CEES14] Louise H. Crockett, Ross A. Elliot, Martin A. Enderwitz, and Robert W. Stewart. *The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC.* Strathclyde Academic Media,, July 2014.

[Chr11] Edward Chrusciel. *SAMBA Technology Backgrounder.* FUJIFILM Dimatix Inc, February 2011.

[FUJ16] FUJIFILM. *PM000081 SAMBA Printhead Product Manual.* FUJIFILM Dimatix Inc, March 2016.

[Hal17] Christian Halbfurter. *Testplatine für High-Speed Datenverarbeitung.* Graz University of Technology, Institute of Technical Informatics, March 2017.

[IMH12] Graham D. Martin Ian M. Hutchings. *Inkjet Technology for Digital Fabrication.* Chichester, West Sussex, United Kingdom, Wiley, November 2012.

[JCW16] Huanhuan Jia, Yunzhi Chen, and Xin Wang. *Application of the New Hybrid Screen Dot in Inkjet Plate-Making Technology*, pages 443–451. Springer Singapore, Singapore, 2016.

[Joh15] Jeff Johnson. *Comparison of Zynq SoMs.* Opsero, May 2015.

[Kal16] Vaibhav Kale. *WP469 Using the MicroBlaze Processor to Accelerate Cost-Sensitive Embedded System Development.* Xilinx Inc., June 2016.

[Lab15] Silicon Laboratories. *Si864x Low-Power Quad-Channel Digital Isolators Data Sheet.* Silicon Laboratories Inc., November 2015.

[Tay17] Adam P. Taylor. *How to use Interrupts on the Zynq SoC.* e2v Technologies, 2017.

[Xil14] Xilinx. *PG172 Integrated Logic Analyzer v5.0 Product Guide.* Xilinx, 2014.

[Xil16a] Xilinx. *UG578 UltraScale Architecture GTY Transceivers.* Xilinx Inc., December 2016.

[Xil16b] Xilinx. *UG585 Zynq-7000 All Programmable SoC Technical Reference Manual.* Xilinx Inc., September 2016.

[Xil17a] Xilinx. *Zynq System on Modules (SoM) boards.* Xilinx, 2017.

[Xil17b] Xilinx. *DS180 7 Series FPGAs Data Sheet: Overview.* Xilinx Inc., March 2017.