Bachelor's Thesis

Development And Programming Of A Data Aquisition Component For The New ATLAS Pixel Detector Front-End Electronics

prepared by

Johannes Agricola

from Fulda

at the II. Physikalischen Institut

Thesis number:	II.Physik-UniGö-BSc-2011/10
Thesis period:	1st April 2011 until 30th September 2011
First referee:	Prof. Dr. Arnulf Quadt
Second referee:	Prof. Dr. Dieter Hogrefe

Abstract

The ATLAS Pixel Detector is planned to be upgraded with the "Insertable B-Layer" (IBL) in 2013. The read-out chain adapted to the new requirements of the new Pixel Detector layer is currently in development. The electrical Back-Of-Crate card designed, configured, successfully simulated and tested in this thesis is going to be used to set up a test system and a Pixel Detector read-out system for the laboratory that comes without optical transmission lines and their problems. It is a redesign of the existing electrical Back-Of-Crate card, adapted to the requirements introduced by the new sensor chips. Especially the 8B/10B coded data transmission and the increased data rate of 160 Mbit/s for a single link from the sensor chip are relevant to this thesis, as any data from the sensor chip had to be decoded, buffered and demultiplexed onto 40 Mbit/s links.

Keywords: Large Hadron Collider, ATLAS, IBL, Detector Read-Out, FE-I4, FPGA

Contents

Nomenclature

Acronyms

Acronym	Meaning
ASIC	Application-Specific Integrated Circuit
ATLAS	A Toroidal LHC Apparatus
BOC	Back-Of-Crate card
CERN	European Organization for Nuclear Research
CLB	Configurable Logic Block
DCM	Digital Clock Manager
DDR	Double Data Rate
DFS	Digital Frequency Synthesizer
DH	Data Header
DLL	Delay-Locked Loop
DR	Data Record
eBOC	electrical Back-Of-Crate card
EOF	End-Of-Frame
FE-I3	Front-End Interface 3
FE-I4	Front-End Interface 4
FIFO	First In First Out
FPGA	Field Programmable Gate Array
IBL	Insertable B-Layer
I/O	Input/Output
JTAG	Joint Test Action Group
LHC	Large Hadron Collider
LUT	Look-Up Table
LVDS	Low-Voltage Differential Signalling
MCC	Module Control Chip
MUX	Multiplexer

Acronym	**Meaning**
PCB	Printed Circuit Board
PECL	Positive Emitter-Coupled Logic
PLL	Phase-Locked Loop
PP0	Patch-Panel 0
PROM	Programmable Read-Only Memory
QFP	Quad Flat Pack
RAM	Random-Access Memory
RD	Running Disparity
ROD	Read-Out Driver
ROM	Read-Only Memory
RTL	Register Transfer Level
SBC	Single Board Computer
SOF	Start-Of-Frame
SR	Service Record
TIM	Timing, Trigger and Control Interface Module
ToT	Time over Threshold
VCO	Voltage Controlled Oscillator
VHDL	Very High Speed Integrated Circuit Hardware Description Language

1. Introduction

In March of 2011 the Large Hadron Collider began operation with proton-proton collisions at 3.5 TeV per beam. Only around five months later, in August, ATLAS and CMS, two of the experiments at the Large Hadron Collider, have collected about 2 fb^{-1} of data[1]. This equates to the amount of data produced in about $140 \cdot 10^{12}$ collisions. To filter this data and save and process only relevant fractions is a complex task which is accomplished by a sophisticated triggering and read-out system.

The innermost part of the ATLAS experiment is the Pixel Detector, a detector that records the energy of charged particles at discrete points on three layers around the collision point. In 2013 a new Pixel Detector layer is planned to be inserted into ATLAS. Due to its closer position to the interaction point and the collider's run parameters in the ensuing operation, it will have a higher resolution. This severely increases the amount of data produced per area and calls for new data transmission techniques and thus an updated detector read-out system.

Together with other improvements this leads to a new front-end electronics with notably increased bandwidth per link and a new data protocol encoded with a transmission code. In addition to this the off-detector electronics are currently updated.

In this thesis an interim solution for the migration to the new read-out electronics is created which is also stripped of the optical transmission features. This electrical Back-Of-Crate card, which is a replacement for the default Back-Of-Crate card, should allow to connect the new front-end electronics to the old read-out electronics.

In Chapter 2, the Large Hadron Collider, the ATLAS experiment, especially its Pixel Detector including the read-out electronics and upgrade plans are introduced. A general understanding of FPGA technology and configuration is established and an overview of the 8B/10B transmission code is put across. The actual contribution of this thesis is described in Chapter 3 where the hardware and its configuration are presented. Chapter 4 contains the results of simulation and testing of the configuration and hardware.

[1] http://cdsweb.cern.ch/record/1372204

2. Background and Related Topics

2.1. The ATLAS Pixel Detector

The Large Hadron Collider (LHC) [1] is a circular accelerator set in a tunnel with 27 km circumference located at CERN (European Organization for Nuclear Research) near Geneva, Switzerland.

Preaccelerated proton or lead packages are injected into the beam pipe at 450 GeV. After further acceleration, proton-proton collisions take place at four points located along the beam pipe, at a center of mass energy of up to 14 TeV. A maximum of 2808 proton packages will cycle the LHC ring at the same time and collide at a rate of up to 40 MHz.

The four aforementioned sections of the LHC ring house the main experiments: ATLAS (A Toroidal LHC Apparatus) and CMS (Compact Muon Solenoid), focusing on physics in the scale of TeV, ALICE (A Large Ion Collider Experiment), focusing on lead-lead collisions, and the LHCb.

2.1.1. The ATLAS Experiment

The ATLAS [2] experiment consists of several sub-detectors that are layered as barrels around the collision point and extended by caps on both ends (see Figure 2.1). Altogether, the experiment has a radius of 11 m and a length of 46 m.

The muon spectrometer is the outermost and largest part. It is set in a magnetic field of 0.5 T to accurately measure the momentum of single muons through their path. The magnetic field is generated by a set of eight toroidal magnets which radially cut the muon spectrometer.

Next to this are the calorimeters: First when looking from the interaction point is the electromagnetic calorimeter (also called Liquid Argon Calorimter due to its sampling material) which measures the energy of electromagnetically interacting particles like charged particles and photons by absorbing them. Particles that pass through it but interact via strong force are then absorbed by the hadronic calorimeter.

Inside the calorimeters, a solenoid coil provides a magnetic field of 2 T for the inner detector which consists of three more detectors: The Transition Radiation Tracker (TRT),

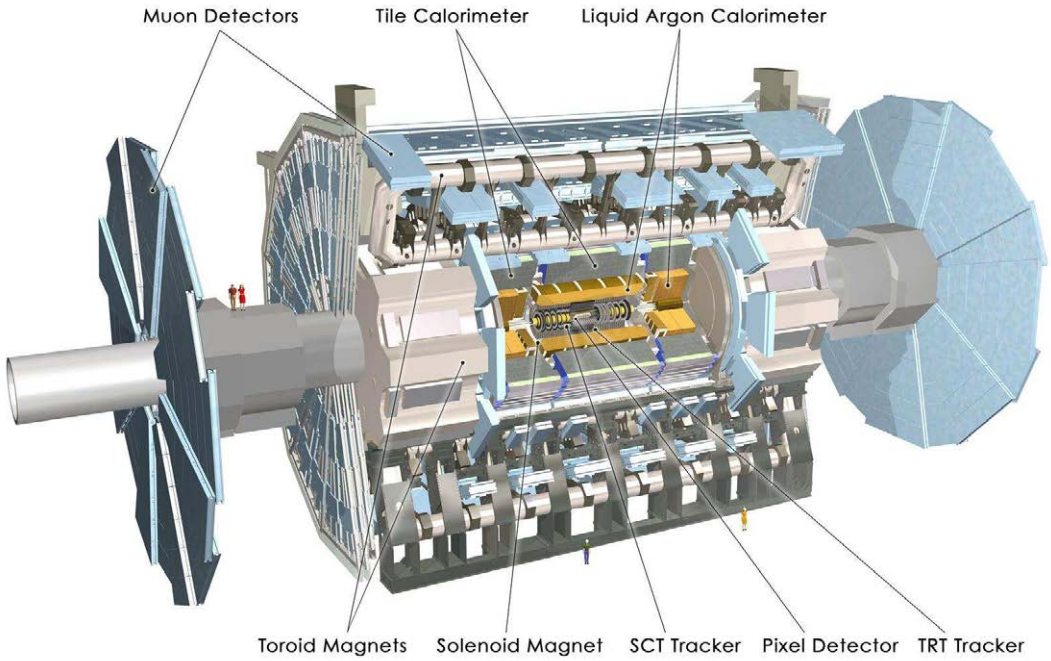

Figure 2.1.: Overview and size comparison of the ATLAS detector experiment.

	Radius r	Number of Modules
B-Layer	50.5 mm	286
Layer 1	88.5 mm	494
Layer 2	122.5 mm	676

Table 2.1.: Distance and module population of the Pixel Detector barrels.

the Semi-Conductor Tracker (SCT) and the Pixel Detector.

The outer detector is the TRT which detects ultrarelativistic particles by their transition radiation at different media. The SCT is basically the same as the Pixel Detector, but with lower resolution and longer pixels.

2.1.2. The ATLAS Pixel Detector

The innermost subdetector, the Pixel Detector is depicted in Figure 2.2. Its outer radius is only 122.5 mm with a length of 1.3 m. It detects charged particles by their interaction with the detector's semiconductor material that is subdivided into units which are appropriately named "pixels".

The detector consists of three barrels (see Table 2.1) with three end caps on each side. As it is covered by the magnetic field generated by the solenoid described above,

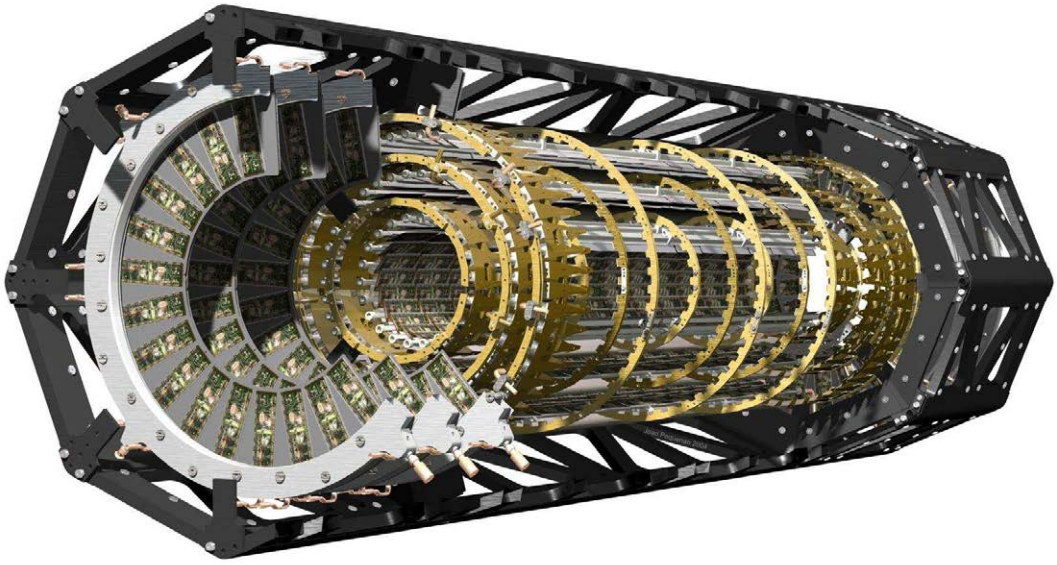

Figure 2.2.: The ATLAS Pixel Detector.

momentum and charge of the detected particles can be reconstructed from their path. It is also possible to calculate their point of origin and thus find out if they are an immediate result of collision at the interaction point or of a subsequent decay.

The described structure is padded with Pixel Modules. Each pixel module hosts a silicon pixel sensor in which the interaction with the charged particle takes place. It is subdivided into 47232 pixels which measure 50×400 µm^2 or 50×600 µm^2 [1].

The front end interface (FE-I3 for version 3) electronics is mounted in a grid of 8×2 chips beneath the sensor. Each of the pixel cells is connected to one analog front end cell on one of the FE-I3s. This electronics amplifies the current pulse from the sensor and compares it to a configurable threshold. The time stamps of the rising and falling edge are then saved for further evaluation: The rising edge is used to match the read data to an event of interest, the difference between rising and falling edge, called the Time over Threshold (ToT), is a measure of the energy the particle lost while traversing the sensor.

All FE chips of a module are connected to one Module Control Chip (MCC) which is responsible for reading out the hits saved on the FE chips, providing clocks, reset signals and communicating with the off-detector electronics. Once a trigger has been received by the MCC, it reads the relevant hits from the FE Chips, aggregates and transmits them to the off-detector electronics.

[1] The larger pixel size is required as the read-out electronics which is described in the following paragraph is divided into several discrete chips and there has to be a safety margin at the border of each die that cannot contain electronics.

One of the characteristics of the MCC that are relevant to this thesis are the supported data rates: At all times, the MCC receives a 40 MHz clock from the off-detector electronics. Synchronized to this clock, each MCC has a command channel over which it receives commands at 40 Mbit/s.

In the other direction, the MCC is able to return data at different rates. First of all, there is the 40 Mbit/s mode where the data is synchronized to every second edge of the clock. Alternatively, a double data rate (DDR) mode is supported where the data is synchronized to every rising and falling edge resulting in a data rate of 80 Mbit/s. Additionally to this, each MCC can be connected with two data lines which implies a maximum data rate of 160 Mbit/s per pixel module with a maximum data rate of 80 Mbit/s per channel.

2.1.3. Insertable B-Layer

As an extension to the three available Pixel Detector layers, a new B-Layer called Insertable B-Layer (IBL) [3] will be inserted into the current B-Layer during the second major LHC shutdown in 2013 [4]. Among other things, this upgrade is motivated by the increasing requirements and stress on the current B-Layer:

First of all, the B-Layer is crucial for vertex reconstruction. Whilst a failure in Layers 1 or 2 is not critical, vertex reconstrcution performance is heavily reduced when a B-Layer module is damaged. Also, precision for other analytical processes is increased by a new detector layer that is close to the interaction point.

Second, the Pixel Detector has been designed for a maximum luminosity of 10^{34} $\mathrm{cm}^{-2}\mathrm{s}^{-1}$. However, due to other upgrades to the LHC the Pixel Detector will be in operation while the peak luminosity reaches twice the maximum value. The current on-detector electronics is unable to handle the number of recorded events which will result in additional inefficiencies.

Other reasons include the preventive replacement of the beam-pipe to avoid damage during detector operation. Also, at the time of planning the IBL project, radiation-induced problems like reduced sensor efficiency were expected. However, the data taken in the first years of LHC operation showed that it is improbable that those are going to be a problem for the doses that are deposited during normal operation.

Of course, as it has to be inserted into the existing structure, the IBL must be mounted at a lower radius, namely at a mean radius of $r = 32$ mm. This has a notable impact on the sensor and electronics design: To achieve the same radial resolution as the existing B-Layer, the resolution has to be enhanced by the same factor as the decrease in distance to the collision point. Also, as a result of the inverse-square law, the increased radiation

exposure calls for a more radiation-hard technology. Due to its position in the detector and the volume restrictions, mass and volume of the B-Layer setup have to be as small as possible. Additionally, the active area of the sensor has been increased.

FE-I4

These parameters turn into a central requirement for the new read-out electronics: A notably increased data rate per area. The different solutions to this requirement culminate in the FE-I4 (Front-End Interface 4) [5], which in contrast to the FE-I3 also takes over the tasks of the MCC.

The main improvement of the digital electronics is a new read-out scheme: Instead of moving hits out of the pixel array every time they are registered, they are saved in memory located directly adjacent to the analog electronics. Then hits are only read out of these cells in case of a trigger.

More relevant to the work performed in this thesis are the modifications to the off-detector read-out interface. Instead of the $2 \cdot 80$ Mbit/s in data rate and one command channel per MCC, each FE-I4 is connected with only one data channel with a data rate of up to 160 Mbit/s and a command channel that can be shared between up to eight FE-I4s due to a sensor addressing scheme that has been introduced.

The data channel is also encoded with the transmission code 8B/10B, which has many desired properties for asynchronous optical or differential data transmission and will be explained in detail in Section 2.3.

2.1.4. Detector Read-Out

The on-detector part of the detector read-out has already been described coarsely in the previous sections. This thesis focuses on the off-detector tasks. In the setup as it is currently used at ATLAS and as shown in Figure 2.3, the Pixel Module is connected to an electrical-optical converter on the detector, called Optoboard. This in turn is connected via optical fibres leading out of the detector to the Back-Of-Crate card (BOC) in the read-out crate. On the BOC optical signals are again converted into electrical signals and vice versa. Depending on the bandwidth, each data channel which is constituted by one or two lines is demultiplexed into up to four lines with a data rate of 40 Mbit/s per line.

These lines lead to the Read-Out Driver (ROD) which is responsible for histogramming and analysis when in calibration mode and for event building when capturing actual data. These events are then transmitted to the Read-Out Buffer for further processing.

Amongst other things, the TIM (Timing, Trigger and Control Interface Module) is also

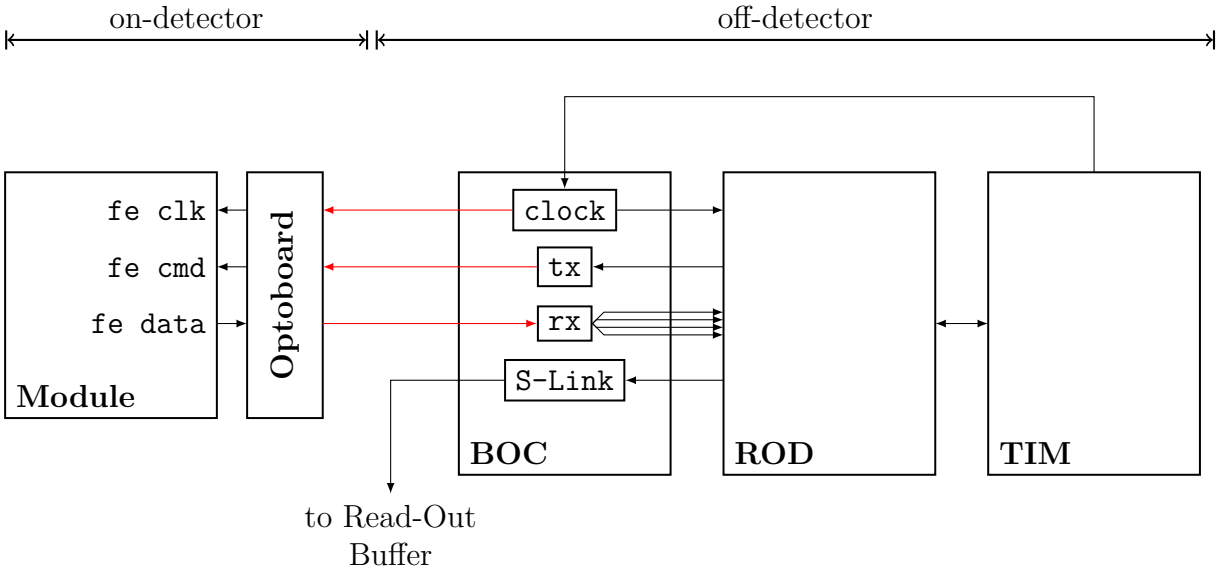

Figure 2.3.: On-site detector read-out scheme. Red lines denote optical, black lines electrical connections.

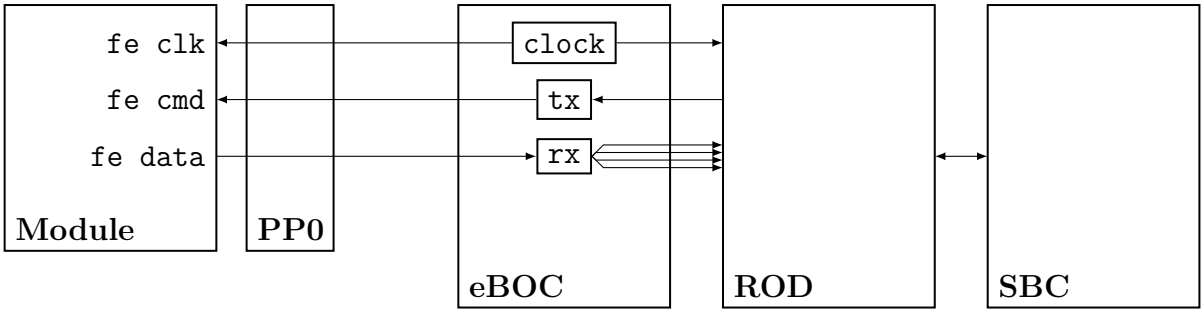

Figure 2.4.: Laboratory detector read-out scheme.

responsible for forwarding a received trigger to the ROD for further distribution to the Pixel Modules. The TIM also supplies the ROD with a clock which is then distributed to the modules through the BOC.

2.1.5. Laboratory Detector Read-Out - The eBOC

As described in the previous section, the on-detector-off-detector transmission is optical. Tuning the optical transceivers is time-consuming and may have to be repeated several times during operation and the transmission is prone to errors.

In contrast to the setup at the LHC, the distance between the detector and the off-detector electronics is short: About 2 meters as compared to 80 meters. Thus in the lab,

the conversion to optical signals is omitted. As this is an integral part of the BOC, this requires a redesign of the BOC: The electrical BOC or eBOC.

Figure 2.4 shows a setup in the lab with an electrical read-out chain. The Optoboard has been removed and the BOC has been replaced by an electrical alternative. Additionally, the triggering has been taken over by the Single Board Computer (SBC) and no TIM is required.

An eBOC already exists for the FE-I3 with corresponding MCC and is described in [6]. However, it suffers from a major problem: The ROD has 96 input channels, but in configuration for the Pixel Detector, it only accepts data on few of them and those cannot be chosen arbitrarily. On the currently existing eBOC, those channels are not chosen correctly. Also, this eBOC requires an additional board, the PP0 (Patch-Panel 0) which is simply an electrical replacement for the Optoboard and as such not required for an implementation relying on electrical communication only.

To migrate the electrical read-out chain to FE-I4, the first component that has to be replaced is the eBOC as in the old revisions it does not support the high data rates of 160 Mbit/s and the ROD does not accept 8B/10B coded data.

2.1.6. MCC Emulator

An emulator for the MCC has been created by Benjamin von Ardenne [7]. This emulator consists of an FPGA configuration which mimics the behaviour of a real Pixel Module that is equipped with a set of FE-I3s and an MCC, but the output channel of it is encoded in 8B/10B, which is not possible with the real Pixel Module. A major limitation is that it only supports a data rate of 40 Mbit/s.

This allows for an independent redesign of the eBOC as testing is possible without depending on the ROD. Afterwards, the old eBOC is simply replaced with the new one. In the further development, a fully tested eBOC is available to migrate the ROD configuration to the new FE-I4 protocol.

2.2. Programmable Logic: FPGAs

A Field Programmable Gate Array (FPGA) is as already stated by the name a device that incorporates configurable logic circuits. The available resources can be divided into several parts.

The most notable ones are on one hand the logic blocks that contain the actual configurable logic. On the other hand these have to be connected which is done by the

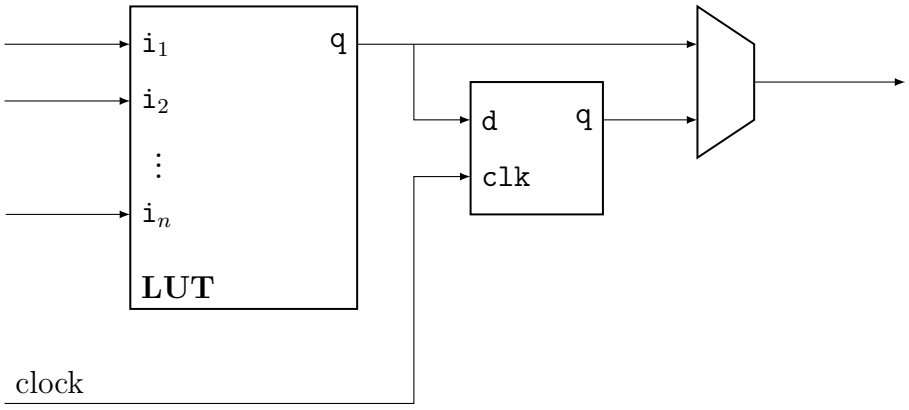

Figure 2.5.: A very simple representation of a CLB.

interconnect matrix. Additional parts may contain resources for clock management, dedicated memory, I/O and digital signal processing.

2.2.1. Logic and Interconnect

The core of an FPGA are its Configurable Logic Blocks (CLBs) or simply logic blocks. A simplified logic block is shown in Figure 2.5: It consists of an n-input Look-Up Table (LUT) and a D-type flip-flop. The LUT can be configured to encode any n-input Boolean function. The flip-flop is used to save the result of this function. Alternatively, the result of the LUT can be directly routed out of the logic block to either implement asynchronous logic or form logic terms with more than n inputs.

Figure 2.5 is simple in comparison to the logic blocks of modern FPGAs [8]. More advanced ones include multiple LUTs, LUTs that are splittable or reconfigurable to RAM or shift registers, logic that enables high-efficiency summation, multiple flip-flops or latches and other features that increase versatility of the programmable circuits.

The other major part of an FPGA are the interconnect resources. Often, they are divided into general-purpose interconnect and the clock network. How these are laid out is vendor and especially device specific. In case of the Spartan 6 architecture [9, 10] used in this thesis, each CLB is connected to a switch matrix (see Figure 2.6). This in turn is connected to many other switch matrices forming the general-purpose interconnect. Additionally, there may be an interconnect responsible for carry logic.

While the general-purpose interconnect is optimized for interconnecting CLBs, often only over a short distance, the clock network [11] is optimized on supplying the whole FPGA with potentially high-speed clocks. Such a clock network is required to be designed for a low clock skew. This is supported by the resources mentioned in the next section.

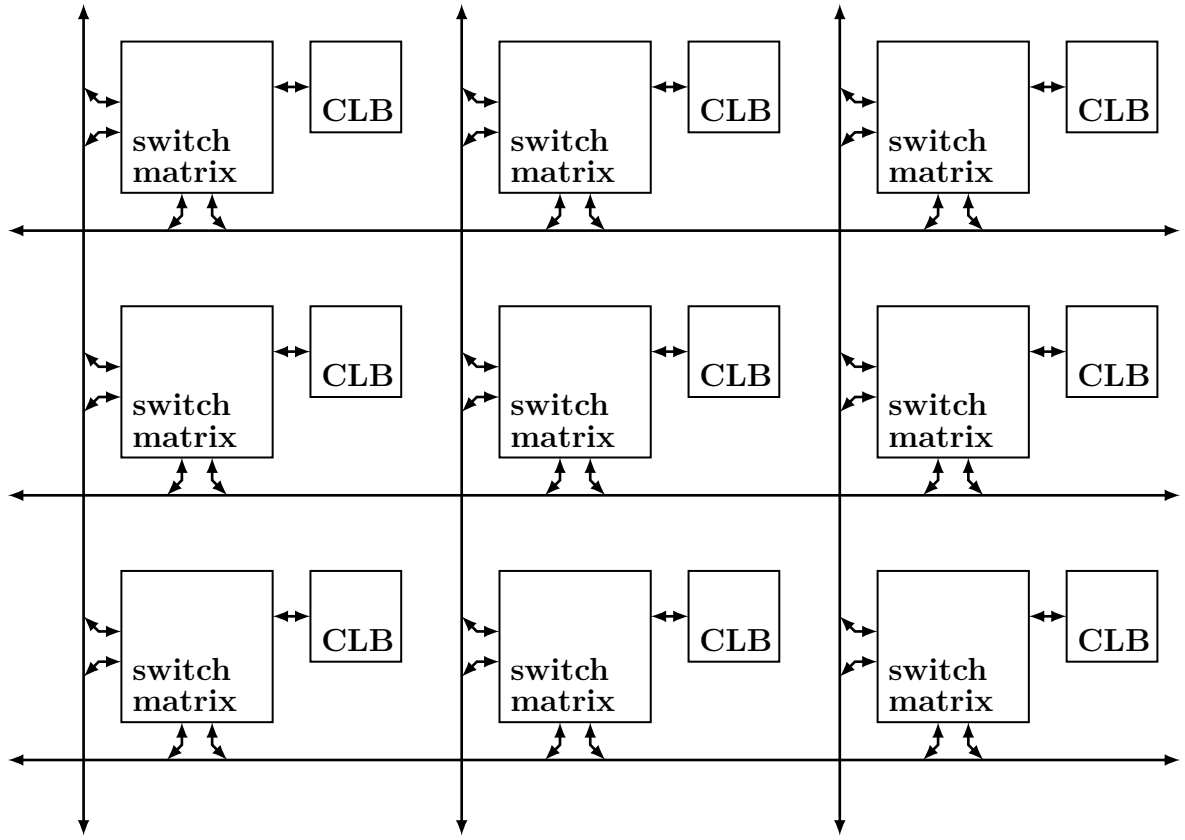

Figure 2.6.: A possible implementation of the general purpose interconnect of an FPGA. Each CLB is connected to its own switch matrix which is able to connect it to horizontal and vertical bus lines.

Often encountered at the edges of the die, the general-purpose interconnect and clock network are connected to special blocks designed to interface to peripheral electronics. These I/O blocks contain specialized electronics for this purpose: Often a multitude of I/O standards is supported, including differential standards. The I/O sections of the FPGA used in this thesis also incorporate serial-parallel-converters which allow for serial communication at a much higher bit rate than possible within the FPGA.

2.2.2. Additional Resources

FPGAs often include additional resources, like phase-locked loops (PLLs) or other clock synthesis and conditioning tools:

The FPGA chosen for this thesis includes several PLLs and digital frequency synthesizers (DFSs). Each PLL has several outputs with different frequency divider and phase shift settings. These are supplemented by delay-locked loops (DLLs) and general-purpose

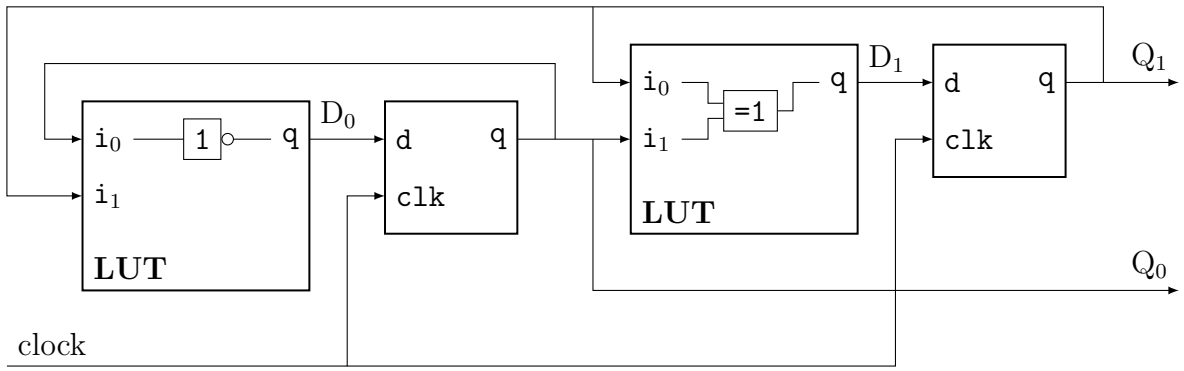

Figure 2.7.: A schematic of a synchronous counter in an RTL view.

phase shifters, which are able to shift the phase of a clock. Specifically, this is used to compensate for the clock distribution delay.

FPGAs are usually ideally suited for high data rates[2]. This defines a requirement for a certain amount of memory, especially with low-latency access within the FPGA.

In FPGA technology, especially on Xilinx FPGAs, one often distinguishes between distributed [11] and block RAM [12]. Distributed RAM refers to the already mentioned reconfiguration of a part of a logic block to RAM. Block RAM on the other hand provides relatively large amounts of RAM in a separate section of the FPGA. It often is dual-ported, reconfigurable to different data widths that may even differ between the two ports and may have specific logic to emulate more advanced memory devices. The contents of the memory may be configurable in band with the rest of the FPGA which enables the memory blocks to emulate fast LUTs with high fan-in and fan-out or ROM.

Additionally it is common to have external memory for an FPGA if necessary. This may be supported through configurable transceivers for those memory chips directly on the FPGA [13] and the support for adapted I/O standards.

2.2.3. The Register Transfer Level

Describing electronic systems can be done on several abstraction layers [14]. The lowest layer is the transistor level: Every circuit is defined directly through the use of transistors (and additional passive components as necessary). Above resides the gate level where logic circuits can be described using logic gates. This is commonly known but does not support abstraction for stateful circuitry: A state would still be required to be represented by a feedback.

[2]in comparison to microcontrollers, for example

```
D0 <= not Q0;                                    1
D1 <= Q0 xor Q1;                                 2

process (clock)                                  4
begin                                            5
    if rising_edge(clock) then                   6
        Q0 <= D0;                                7
        Q1 <= D1;                                8
    end if;                                      9
end process;                                     10
```

Listing 2.1: Example for a circuit description in VHDL on the RTL.

The abstraction layer used to implement the FPGA configuration in this thesis is the register transfer level (RTL). This is common when working with higher-level hardware description languages. Also, the FPGAs are often designed to make working on this level easy: In the RTL, a synchronous digital circuit is described by the logical operations and the registers in which their outcome is saved. This can be easily mapped to the CLB model showed in Section 2.2.1.

Figure 2.7 show an example of a synchronous circuit. There are two D-type flip-flops which represent the registers and two LUTs which implement the combinatorial logic. On each rising edge of the clock, the value of the output of the LUTs is written into the registers. The output value of the registers are then used to calculate the values which have to be written into the registers on the next clock edge. This example could be used to implement any two-bit synchronous counter, depending solely on the LUT configuration.

The figure also shows an example configuration for the LUTs resulting in the following behaviour: On every rising clock edge, the left-hand flip-flop is toggling. The right-hand flip-flop only toggles every second time (think of the XOR as a controllable inverter in this case), resulting in a 2-bit up-counter.

2.2.4. Very High Speed Integrated Circuit Hardware Description Language

The main language used to implement the eBOC configuration is the Very High Speed Integrated Circuit Hardware Description Language (VHDL). Its original purpose was design and simulation of application-specific integrated circuits (ASICs) but is now widely used in FPGA implementation.

Describing the RTL in VHDL is a common task [15]. The same counter as shown in

```
uut: counter port map (                                    1
       clock => clock,                                     2
       q0 => q0,                                           3
       q1 => q1                                            4
   );                                                      5

clock_process: process                                     7
begin                                                      8
   clock <= '0';                                           9
   wait for clock_period/2;                                10
   clock <= '1';                                           11
   wait for clock_period/2;                                12
end process;                                               13
```

Listing 2.2: A behavioural simulation description for a counter.

the previous section is implemented in Listing 2.1. Lines 1 and 2 show the combinatorial logic which is encoded in the LUTs: Line 1 corresponds to the left LUT, line 2 to the right LUT.

The other lines define a logic circuit that is first of all sensitive on changes on the clock line (line 4), specifically only on the rising edges of this clock (line 6). Lines 7 and 8 describe what should happen once a rising edge on the clock has been detected, which is to copy the result of a logic function, D0 and D1, into the registered values Q0 and Q1.

Though it is possible to model much more than only the RTL in VHDL and there are further concepts specialized on FPGA configuration, this is the core of many configurations. Another concept that is supported by VHDL and used extensively in this thesis is the ability to implement a hierarchy: It is possible to divide the configuration for an FPGA into sub-circuits which then can be put together at several hierarchical levels.

2.2.5. Design Analysis

Describing the behaviour of an electronic component on the RTL is not intuitive and conflicts with the perception of more common programming paradigms like imperative programming. To support a designer in his tasks, a set of tools exist to check the functionality of a logic description without having to actually implement it on a physical device.

For one there are simulators [16] which are able to show the design's behaviour in detail. To use this feature, a designer creates a simulation description which, in contrast to the circuit description itself, does not necessarily have to be on the RTL but may contain

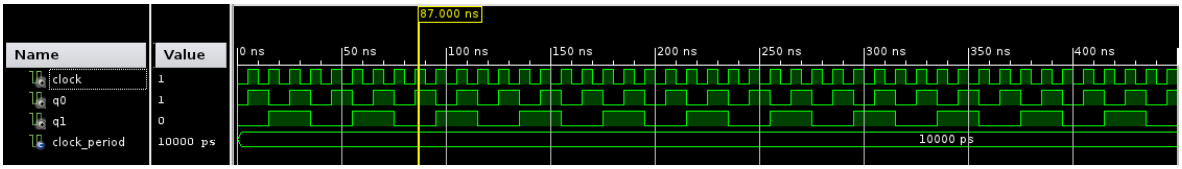

Figure 2.8.: The result of the simulation described in Listing 2.2 (Screenshot).

behavioural elements.

Listing 2.2 shows an example of a simulation description for the counter created in the previous sections. The first block (lines 1 to 5) makes use of the hierarchical features of VHDL and instantiates the counter. Lines 2 to 4 assign the input and output ports of the counter (which were not explicitly shown in Listing 2.1 for simplicity) to local signals (whose declaration is hidden, too). The rest of the code defines a process which is repeated until the simulation halts. It is a behavioural description of the clock which is the only input to the counter. In contrast to an RTL description this contains actual delay times in the *wait for* statements. This is easier to conceive, but generally impossible to map to a common FPGA.

The result of this simulation can be examined in Figure 2.8. It shows the clock, q0 and q1 which behave as expected.

However, a successful behavioural simulation is not sufficient to guarantee a functional design. Though often neglected when first implementing a logic circuit, logic gates and paths introduce measurable delays into a circuit. Also, synchronous logic requires setup and hold delays for signals due to the internal structure of flip-flops.

To check if a design that has been completely mapped to a certain FPGA still behaves as the result of the behavioural simulation, a timing analysis tool [17] can be used. During the design of the FPGA, the vendor characterizes the different components and delays between them. This data is then combined with information on clock speeds to infer that the design can be correctly implemented.

2.2.6. Cores

The toolchains used to configure FPGAs often include soft cores or core generators[3]. These cores can be compared to a library in software design and provide preconfigured logic with certain functionalities.

In this thesis, a FIFO core (see Figure 2.9) generated by a core generator provided

[3]Cores are also referred to as IP (Intellectual Property) cores due to the rights that a vendor might hold.

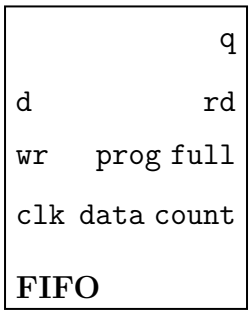

Figure 2.9.: The ports of a FIFO core. This one has a common clock a interface for the amount of data currently saved and a programmable full threshold output.

by Xilinx [18] has been used extensively. The generator supports distributed and block RAM to store FIFO data, different read and write clocks and many other features. The generated FIFO in this example only uses a data count output that signals the amount of words that are currently available for reading and a threshold comparator output on exactly this amount.

2.2.7. Configuration and Debugging

FPGAs usually are configurable and provide a debugging interface via JTAG (Joint Test Action Group) which is a serial bus which was originally designed for debugging the circuits around ICs rather than the ICs themselves.

The FPGA chosen in this thesis also supports configuration through a multitude of other interfaces. One example are the flash PROMs [19] from the same vendor which are able to configure the FPGA using a serial interface. The PROM itself is also configured using JTAG.

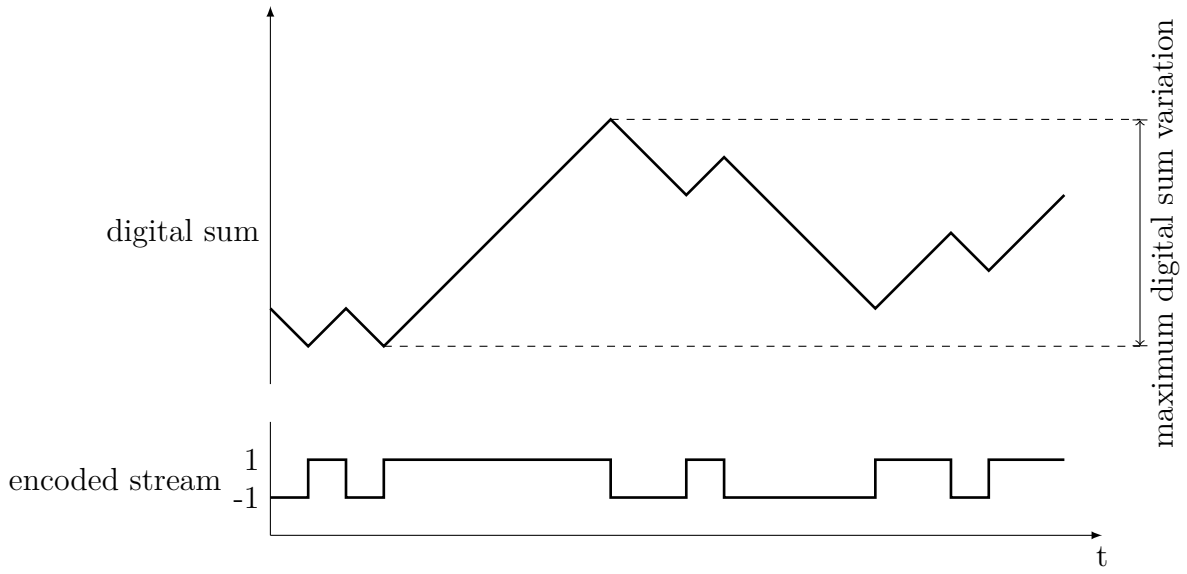

Figure 2.10.: Example data stream with digital sum and maximum digital sum variation for $0 \mapsto -1$, $1 \mapsto 1$.

2.3. 8B/10B Code

2.3.1. Theory

To properly explain 8B/10B, some basic terms from coding theory and general electronics have to be defined first.

The maximum run length of a binary code is the number of consecutive zeroes or ones in any encoded data stream. Thus it is the maximum amount of transmitted data bits between two state transitions in the stream.

The digital sum variation refers to the near-DC energy of the resulting waveform. For binary codes this is usually done by assigning a scalar value to each of the two states, for example $0 \mapsto -1$, $1 \mapsto 1$. Then the digital sum variation is defined as the difference in the sum of these values over a range of an encoded data stream (see Figure 2.10). The maximum digital sum variation of a code is the maximum value the digital sum variation takes over any range of any encoded data stream.

The DC-offset of a waveform is its mean value. A waveform is DC-balanced if its DC-offset is zero. A code which is able to produce a DC-balanced waveform given appropriate transceiver electronics is also called DC-balanced.

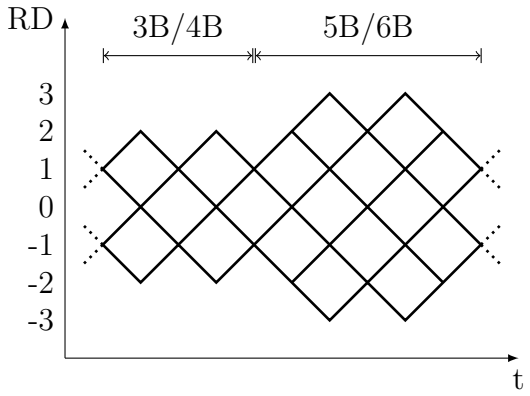

Figure 2.11.: The possible ways the disparity of a single code word in the digital sum space.

2.3.2. 8B/10B

8B/10B is a transmission code that encodes 8 bits of data into a 10-bit wide symbol whilst maintaining DC balance, a maximum run length of 5 and a maximum digital sum variation of 6 [20]. DC balance and a small run length and maximum digital sum variation are properties required for AC coupling of the signal which in turn is helpful for optical transmission.

To achieve these properties, an 8B/10B encoder always keeps record of the digital sum variation in reference to the beginning of the transmission with an offset of 1 or -1. This value is called running disparity (RD).

The code is then restricted so that the RD has to be 1 or -1 at the end of each code word. This implies that only code words that have a digital sum of -2, 0 or 2 are allowed. Additionally, the RD has to stay within $[-3, 3]$ when measuring after every bit. These restrictions are tightened as the encoder is divided into a 3B/4B and 5B/6B encoder and both sub-words have to comply with them, too. Also, the 3B/4B block keeps the RD within $[-2, 2]$ [4].

The resulting possible paths an encoded word can follow in the space of the RD over time are shown in Figure 2.11. The previously stated properties can be easily inferred from here. A complete coding map is included in the appendix of this document.

The mentioned restrictions severely limit the number of code words that are valid out of the codomain $\mathbb{B}^{10}$. Also, it is clear that if one allows the RD to alternate between -1

[4]These restrictions are alleviated in one case as it would be impossible to implement the 3B/4B code else. In this case three consecutive ones or zeroes are allowed which results in an RD at the end of the 3B/4B code that is neither -1 nor 1. However, two alternate encoding forms are offered for this symbol to avoid running into more than 5 consecutive zeroes or ones.

Function	Name	Decoded Representation	Encoded Representation	
		k = 1	RD = −1	RD = +1
Sync	K.28.1	00111100	001111 1001	110000 0110
EOF	K.28.5	10111100	001111 1010	110000 0101
SOF	K.28.7	11111100	001111 1000	110000 0111

Table 2.2.: Comma symbols of special interest.

and 1 and the RD of a code word is not zero, it is required to supply two code words for the same source symbol, one with positive and one with negative RD to allow the encoder to choose one of them depending on the RD before that symbol.

Though there are still some code words unused: These are used as special characters in the data stream and for in-band signalling for special functions. A special subset of these symbols are the *comma* symbols.

The characteristic that discriminates a comma symbol is that its bit sequence is unique in the whole data stream. Thus it is possible for a receiver that receives a comma symbol to exactly determine the symbol boundaries of the comma and all subsequent symbols. There are three comma symbols available and in general they are assigned the function Start-Of-Frame (SOF), End-Of-Frame (EOF) and a synchronization symbol that can be transferred if no data is available to enable the receiver to recover the sender's clock. As these sequences are especially interesting, they are listed in Table 2.2.

3. Implementation

The aim of this thesis is to build a new electrical Back-Of-Crate card adapted to the new requirements coming with the FE-I4 as described in Section 2.1.3. It has to connect to the correct input links of the ROD and decode the 8B/10B in the data channel from the FE-I4. Ideally the implementation should no longer require the PP0 board.

It was given that at least six FE-I4s should be handled by the eBOC at the same time at full speed. Due to the chosen hardware, it was easily possible to go with eight FE-I4s at 160 Mbit/s, which also is the maximum the current ROD can handle at that data rate.

This section is organized in the same order as the actual implementation has been performed. First, the whole eBOC configuration was defined and simulated, with the latter being described in the next chapter. The configuration has then been analysed and the results were used to design the hardware, to which the configuration has then been mapped.

This document will often refer to the available data rates, which always are one, two or four times the base clock with a frequency of 40 MHz. Thus, in the remaining of this thesis, r will always denote the common rate multipliers 1, 2 or 4.

3.1. Configuration

An overview of the eBOC configuration is given in Figure 3.1. It can be divided into four separate parts: The transmitter section handles the forwarding of commands from the ROD and the receiver section is responsible for processing of the data channel from the FE chips. One receiver-transmitter pair handles one FE chip. Additionally, there are blocks responsible for clock synthesis as well as configuration and status management.

3.1.1. Data Path

The most complex of the four sections is the receiver. It takes an asynchronous data stream at up to 160 Mbit/s that has been encoded with 8B/10B from the FE chips. The following section describes one of the eight equal receivers.

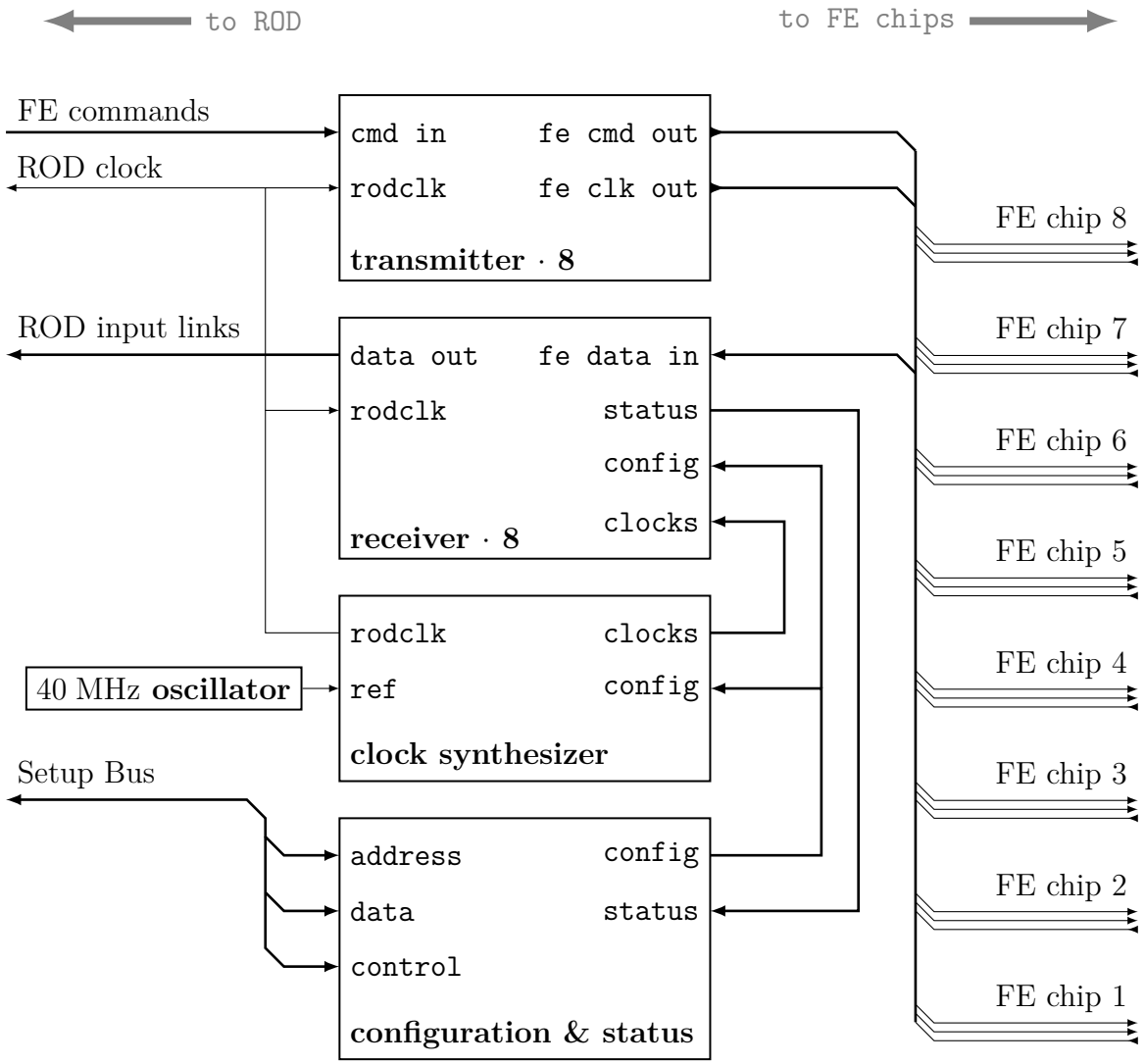

Figure 3.1.: eBOC configuration overview: The eight transmitters are responsible for forwarding the commands from the ROD to the FE chips synchronously to a supplied clock. The receivers in turn read an asynchronous transmission from the FE chips and forward them to the ROD. The clock synthesizer takes a 40 MHz reference clock and generates several clocks required for the receivers, transmitters and the ROD. Configuration & status provides the configuration to the receivers and clock synthesis modules and is able to forward the state of the receiver to the ROD.

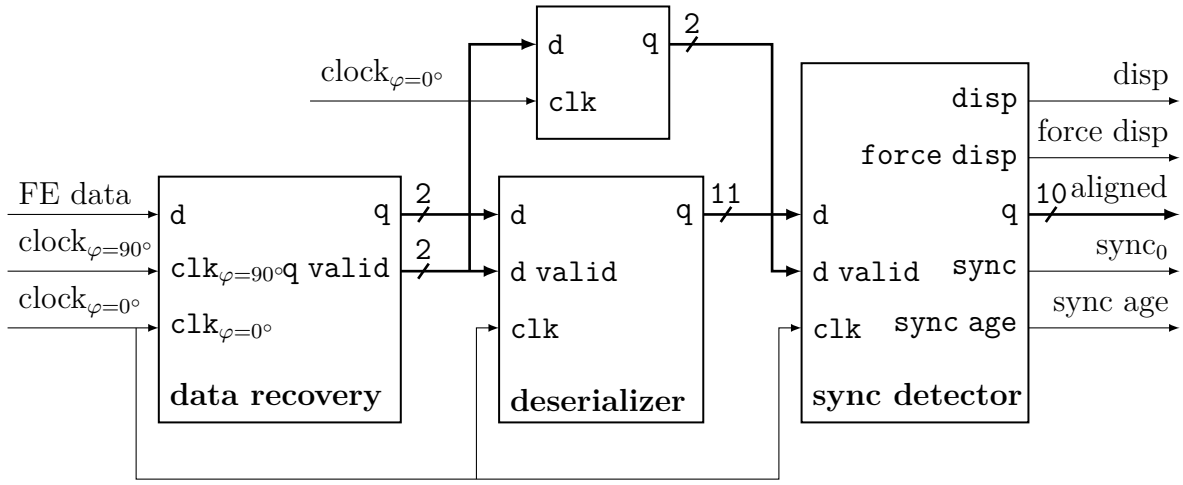

Figure 3.2.: Receiver, input stage: The data is synchronized, deserialized and aligned to 8B/10B symbols.

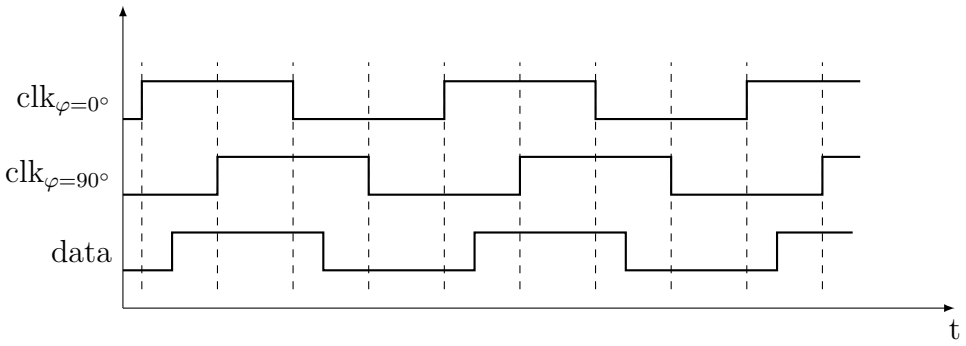

Figure 3.3.: Four-times oversampling with two clocks at the same frequency. Sampling points are marked with a dashed line.

In a first step (see Figure 3.2) this stream is resynchronized using the approach described in [21]. The general idea behind this *data recovery* process can be divided into two techniques: First, four-times oversampling through operating with a low system frequency is achieved by sampling the data on the rising and falling clock edge and doing the same using a clock that is shifted by 90° (see Figure 3.3). Second, edges on the data stream are tracked and the resulting information is used to select the best sampling point. By moving this sampling point, differences in the frequency between sender and receiver are compensated.

However, moving the sampling point on data that is received and processed at the same frequency results in situations where it is possible that on a clock edge of clock$_{\varphi=0^\circ}$, which will be used for clocking the following logic, zero or two bits are available instead of one.

This has to be taken care of in the next steps.

The subsequent *deserializer* follows the data recovery and is connected to it via two 2-bit wide buses, two data bits (`q`) and two bits signalling which of the data bits is valid (`q valid`). The deserializer is a serial-in parallel-out shift register, which, in contrast to a common shift register, is able to shift by two bits in one clock cycle to accommodate the data recovery behaviour. Though each 8B/10B symbol has a width of 10 bits, the output of the deserializer `q` has a width of 11 bits. This is due to the fact that if two bits are shifted in at the same time, the upper or lower 10 bits of the output may contain the symbol and have to be analyzed separately.

The next link in the chain is the *sync detector*: Its main responsibility is to correctly align the subsequent operations to the 8B/10B symbols on the bitstream. This is done by looking for the synchronization symbol in the last 11 bits that were received. Once such a symbol has been detected, the `sync` output is asserted for one clock cycle and the respective 10 bit wide range of the aforementioned shift register is latched through to the wire *aligned*.

The sync detector also keeps a counter of the clock cycles since the last sync and repeats asserting `sync` every ten valid bits. This means that the data recovery output `q valid`, specifically a delayed version of it, to resynchronize it with the data from the deserializer, is taken into account to either expedite or defer this sync pulse and again to choose between the upper or lower 10 bits of the deserializer. Additionally, a counter which holds the number of 8B/10B symbols since the last sync is provided to the reset circuitry.

At this point, the input stage of the data path is nearly complete. The following logic will only see a correctly aligned 8B/10B symbol once the sync signal is high. One piece however is still missing. For the decoder to be able to correctly decode the aligned symbol, it has to start at the same RD as the encoder. Also, in case of an error, a correct RD has to be recovered. Thus the sync detector uses the disparity of the synchronization symbol to force the 8B/10B decoder RD to the correct value every time such a symbol is received. This is possible as the synchronization symbol has a nonzero RD and thus the RD afterwards is always defined without the necessity of knowledge about prior symbols.

Next the previously aligned symbol is decoded by an 8B/10B *decoder* (see Figure 3.4) provided by Xilinx [22]. Every time the $sync_0$ wire is high, the output of this decoder is either the decoded byte if `k` $= 0$ or a representation of a control symbol if `k` $= 1$. Also, the decoder outputs the RD after the last decoded symbol (`disp out`). While decoding data, this is fed back into the RD input `disp in` to allow normal operation of the decoder. When receiving synchronization symbols, the `force disp` output of the sync detector

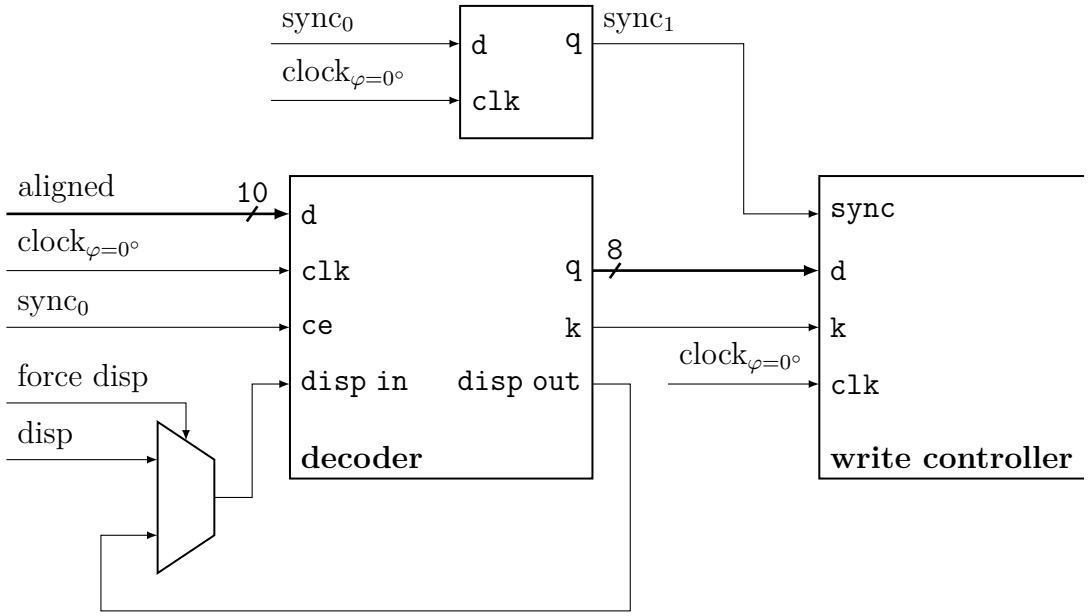

Figure 3.4.: Receiver, decoder stage: The aligned symbols are decoded, frame detection is performed in the write controller.

switches the multiplexer to its own RD output. This implements the already mentioned RD override.

As this decoder delays the output by one clock cycle, the sync signal is delayed, too. These signals, the delayed sync $sync_0$, the decoder data output and the decoder command flag `k` are then connected to the *write controller*. Its purpose is to lock onto data frames: It looks for Start-Of-Frame (SOF) and End-Of-Frame (EOF) control symbols. Once an SOF has been read, all following data bytes are written into the subsequent *buffer FIFO* (see Figure 3.5). Once an EOF has been found, the write controller stops to write data and in turn asserts the `flush` output for one clock cycle.

On the other side of the buffer, the *output controller* waits for this `flush` signal. Once it goes high, the value of the `data count` output of the FIFO, which represents the number of bytes currently stored in it, is copied into a local counter. The FIFO is then read and is decreased simultaneously until it has reached 0. Thus the part of the FIFO that contains fully transferred frames is completely read.

The necessity to buffer the data instead of directly writing it to the ROD after decoding evolves out of a mismatch in the net data rates of the input and output data stream. An FE chip is able to transmit at $r \cdot 40$ Mbit/s. This transmission is 8B/10B coded which yields only eight data bits per ten transmitted bits. Thus the net data rate of the input stream is $r \cdot 32$ Mbit/s. The ROD on the other hand can only accept data streams

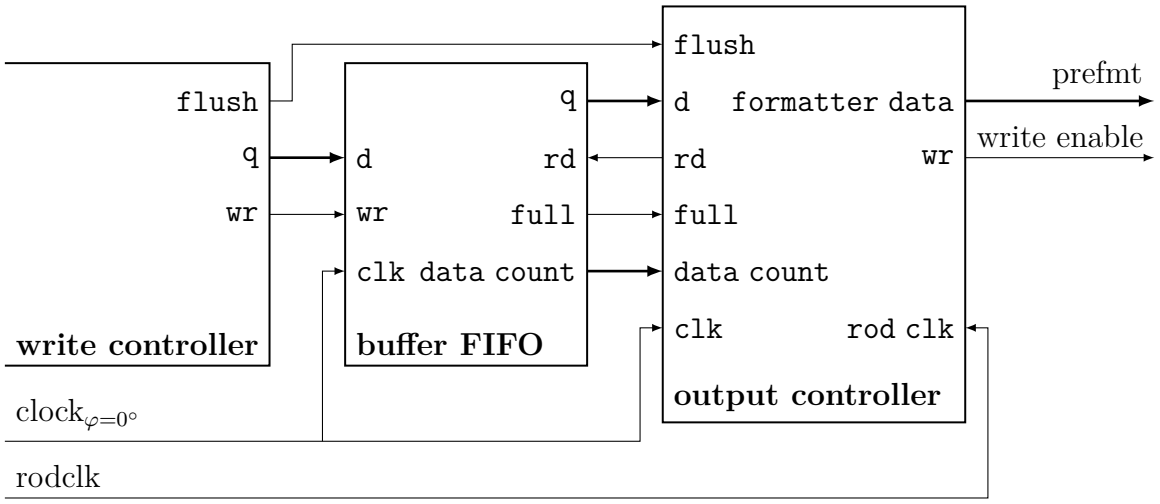

Figure 3.5.: Receiver, buffer stage: Frames are buffered and forwarded to output stage.

at $r \cdot 40$ Mbit/s. As frames have to be forwarded as a whole, the differences in these bandwidths have to be compensated by the buffer. To add 1/5 the output rate to the input rate for the duration of one frame, it is necessary to buffer at least 1/5 of the frame.

In general however, it is not possible to predict the size of a frame. Thus a frame would have to be buffered completely, with one exception: It is possible to calculate the maximum frame size that an FE chip will produce. Once the FIFO contains 1/5 of this size plus a small safety range for frequency deviations of the clocks and to avoid the need to check for off-by-one errors in the configuration, a flush can be safely initiated.

The maximum frame size for FE-I4 is easily calculable: First of all it is clear that only the *Pixel Data* record sequence can produce the biggest frame size. It consists of:

$$\text{SOF},\ (1) \times \text{DH},\ (0..n) \times \text{DR},\ (0..1) \times \text{SR},\ \text{EOF}$$

with n given by the amount of pixels triggered. [5].

The SOF and EOF are not transmitted to the ROD and therefore remain uncounted. All records, the Data Header (DH), containing meta data about the following records that are used to map the data to an event, Data Record (DR), which contains the actual data, and Service Record (SR), which is used for error signalling, each are 3 bytes long. Each DR contains 2 pixel values. If all $80 \cdot 336$ pixels are triggered, 13340 DRs are in the respective frame. Thus the maximum size is

$$s_{\max} = (n_{\max} + 2)\ \text{records} \cdot 3\ \frac{\text{Bytes}}{\text{record}} = 40326\ \text{Bytes}$$

Therefore, a manual flush could be initiated after 8065 B. This is convenient, as a FIFO which contains 2^{13} Bytes = 8192 Bytes can be chosen and its `full` flag can be used

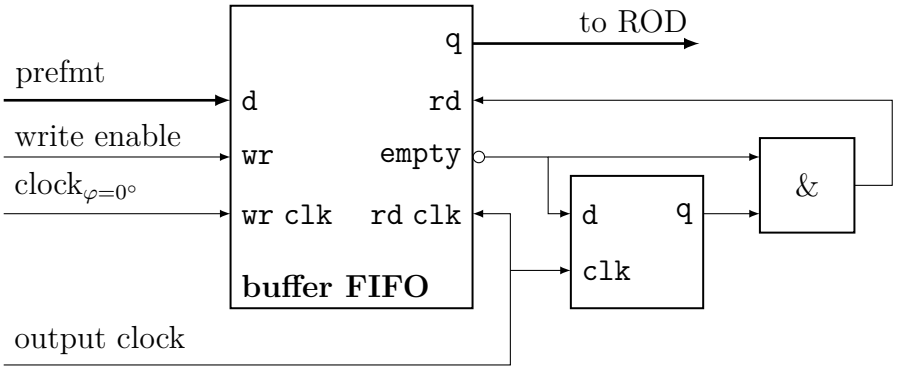

Figure 3.6.: Receiver, output stage: The data bitstream is synchronized to the output clock and then forwarded to the ROD.

to signal a flush without EOF symbol.

The flushing itself, though synchronous to the clock of the rest of the receiver, has to be performed at 40 MHz as required by the ROD. This is implemented by skipping none, every second (for 80 Mbit/s) or every second and third (for 160 Mbit/s) clock cycle, depending on the receiver's clock rate.

To synchronize the data to a clock with a known phase relationship to the ROD clock (see next Section) a FIFO is used (see Figure 3.6). It has separate read and write clocks, the write clock is the 0° clock of the receiver, the read clock is the output clock. Write is always enabled at the same clock edges the output controller did not skip. The read enable input `rd` of the FIFO is connected to its `empty` signal in a way that guarantees that the FIFO will always contain at least one readable bit at a rising output clock if it stays in a stable phase relationship with clock$_{\varphi=0^\circ}$. If this is not the case, for example when skipping a clock edge while switching between data rates, it will stretch the last bit and reestablish the buffer fill.

3.1.2. Clocks

Clock synthesis and distribution is not nearly as complex as the receiver, but it still holds a concept that on the one hand reduces the amount of required logic in the eBOC and on the other hand makes working with the eBOC simpler, as no configuration changes for different laboratory set-ups are necessary.

To get an overview of the required clocks, one first has to find out where data is sent synchronously to which clock. First of all, there is data transmitted from the eBOC to the ROD. As the eBOC is also the clock source for the ROD, synchronization is achieved by phase-shifting the latching clock, called output clock, in reference to the clock transmitted

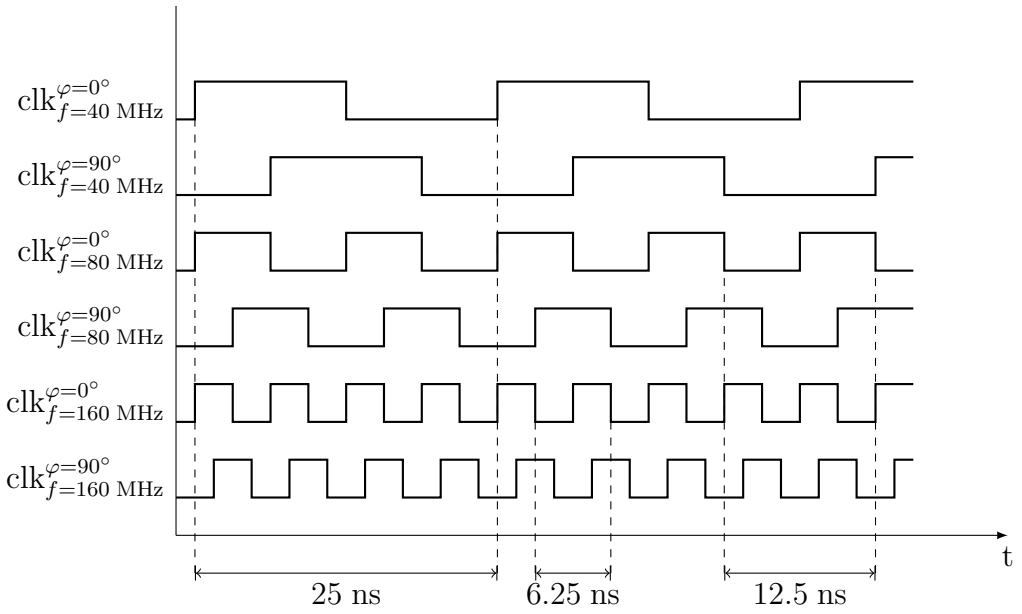

Figure 3.7.: Phase and frequency relationships of the eBOC receiver clocks.

to the ROD, called ROD clock, so the ROD samples the data near the optimal point of the waveform. Back from the ROD return the command lines. Again, another clock is generated and used internally to correctly sample this waveform, referenced as FE clock. The commands are then forwarded to the FE chips which can happen synchronous to the FE clock. The loop is closed with the data returning from the FE-I4. However, it is handled asynchronously to avoid the need to manually compensate for different cable lengths.

As a result of this clocking scheme, it is possible to choose different cable lengths or use the FE-I4 at different temperatures without having to adjust anything on the eBOC. Also, no resynchronization has to be performed, except for the data recovery.

The phase relationships of the FE clock and the ROD clock to the output clock $\varphi_\mathrm{c} - \varphi_\mathrm{b}$ and $\varphi_\mathrm{d} - \varphi_\mathrm{b}$ are constant and can be determined empirically and then set during design time of the eBOC. It mainly depends on the short and fixed wire lengths between the eBOC and the ROD, the fixed transceiver types and the temperature of the electronics, which, in a cooled crate, is held within a small range, too.

In addition to this, the eBOC needs a variety of clocks at different frequencies and phase relationships (see Figure 3.7). For processing of the $r \cdot 40$ Mbit/s data input, clocks at 40 MHz, 80 MHz and 160 MHz are necessary. Additionally, another three clocks at the same frequency and shifted by 90° are required for the data recovery process.

The latter six clocks are generated by one PLL (see Figure 3.8) based on a 40 MHz

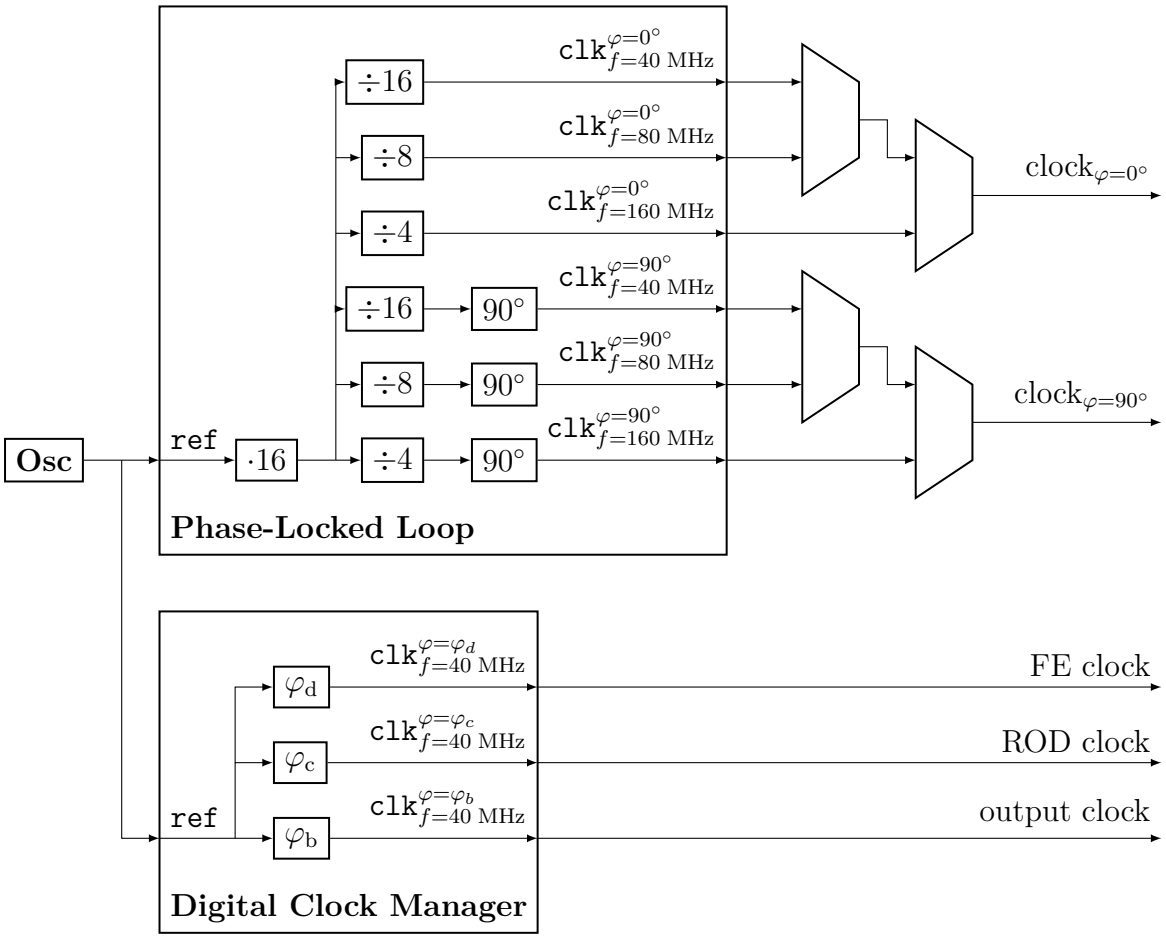

Figure 3.8.: Clock synthesis: The PLL generates the receiver clocks, the DCM is responsible for phase shifting sampling and output clocks.

reference clock. This clock is first multiplied by 16 which is required to stay within the operating range of the Voltage Controlled Oscillator (VCO) which actually generates this 640 MHz signal. The frequency of this signal is then divided down by 16, 8 or 4. The VCO has outputs with 0°, 45°, 90°, ... phase shift. Those are used to generate the phase-shifted clocks. At the output of the PLL, the correct clock for the current mode of operation is selected using multiplexers that are designed to switch between clocks, called BUFGMUX.

The first three clocks are generated by a DLL[1] of the FPGA (also Figure 3.8). It inserts an appropriate delay into the clock lines to shift the clock to the ROD by φ_c and the clock for sampling data from the ROD by φ_d.

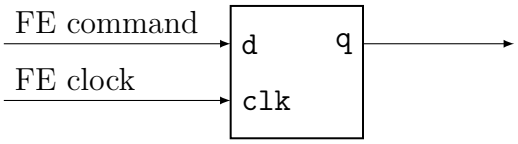

Figure 3.9.: Scheme of the command path.

3.1.3. Command Path

In the command path the ROD transmits its commands to the FE chip. Due to the clocking scheme, the tasks of the FPGA in this path are trivial by comparison to the receiver path. It simply registers the incoming data stream to match the clock phase of the FE chip clock (see Figure 3.9). Afterwards it is directly forwarded to the FE chips.

3.1.4. Setup Bus

The FE-I3 + MCC version of the eBOC presented in Section 2.1.5 did not have to perform complex tasks. Especially, except its initial configuration with DIP switches[2]and the clock management, it did not contain states. The receiver of the eBOC for FE-I4 inherently has several states which may be of interest to the rest of the read-out electronics. This begins at the input stage where the data recovery has to hold information on where to sample the incoming data stream and the sync detector which has to be able to keep the data stream aligned to symbols without any commas for a long period of time. Further into the receiver are the 8B/10B decoder which holds the RD and is able to detect some errors and the buffer which in normal operation at times holds a full frame.

For the BOC, a separate band, the Setup Bus is used to configure and read its state. The eBOC for FE-I4 is planned to use this bus as for configuration (as alternative to the switches on the board itself) and to send back state information. This, however, is not yet implemented. For now, the eBOC is configured with 8 control lines which are wired to DIP switches.

3.1.5. Reset Circuitry

The eBOC does not contain a global reset circuit. Its initial state is established by the fixed FPGA reset during configuration. All external components are stateless and thus do not require a reset.

[1]which is contained in a Digital Clock Manager
[2]DIP switches are switches specialized on configuring electronics equipment. Their name is derived from the Dual Inline Package.

3.2. Hardware

3.2.1. FPGA Selection

Before a new eBOC could be designed, an appropriate FPGA had to be chosen. There is a set of definite criteria for this selection:

Of course, the FPGA has to be able to hold the logic described in the last section, preferably with some additional space for future modifications. The device occupancy in terms of logic can be determined running the implementation tools of any considered FPGA to map the design to it. This criterion turned out not to be a problem, as the design is able to be synthesized into a comparably small logic complex.

Second, the FPGA has to provide the required number of I/Os that support the required I/O standards[3]. Each FE chip takes up three differential I/Os (clock, commands and data) and five CMOS I/Os[4] for the connection to the ROD. Also, I/Os for clocks and configuration are required. Limitations placed on the same I/O standards have to be taken into consideration. For example, some I/O pins may not support a standard or allow only inputs for a standard on them and restrictions on simultaneously switching outputs (SSOs) may be imposed.

The design runs at clock speeds of up to 160 MHz. Some parts of the circuit even have to be able to be clocked at 4/3 times that frequency for the data recovery to work. Determining an FPGA which is able to support this design can also be done with the implementation tools, specifically the timing analyzer described in Section 2.2.5.

As already mentioned previously in this chapter, the configuration requires a certain amount of memory to buffer at least one fifth of the data record frames. This memory could be external to the FPGA, however, this would require more pins for the memory connection and more logic for the memory controller or a dedicated memory controller.

A fifth criterion is the available package. In general, FPGAs are high pin-density devices that come in appropriate packages like Ball Grid Arrays (BGA). Simple packages with leads that are reachable after the soldering process like Quad Flat Pack (QFP) are becoming more and more an exception in modern FPGA technology. Having to use BGA or even smaller and more complex packages to work with has a number of disadvantages: First, a PCB for pins that are arranged in a grid array with a small pitch has to have

[3] An I/O standard refers to the requirements on receivers and transmitters which should be connected, specifically voltage levels, current or termination properties. Adherence to the I/O standard of a transmitter or receiver is important for correct discrimination between the signalled states.

[4] CMOS (Complementary Metal-Oxide-Semiconductor) refers to a commonly used semiconductor technology. A CMOS I/O is a single ended receiver or transmitter which adheres to the requirements given by this technology.

several layers, depending on the grid size which greatly increases its cost while making it hard to debug. Second, the soldering process for BGA components requires some expertise and equipment, for example an X-Ray apparatus for checking the result.

Additionally, FPGAs may require several voltages for operation. This ranges from one voltage for the whole FPGA to a separate core voltage, separate voltages for the different preconfigured parts of the FPGA like transceivers and memory and of course several voltages for the different I/O standards. With the necessity of more voltages, the PCB routing grows more complex and additional DC to DC converters are required.

A minor criterion that was taken into consideration was the vendor. The FPGAs used on the ROD are made by Xilinx and appropriate tools, software and experience are available.

At last the price of the FPGA has to be taken into consideration. This can range from as little as a few Euros up to several ten thousands which would not be within the budget for this project.

With the restrictions of searching a Xilinx FPGA that has to be available in a QFP package, the originally vast number of devices was reduced severely. When looking at the families still in production, only the low-cost families Spartan 3, Spartan 3A, Spartan 3E and Spartan 6 FPGAs match. Modern Virtex devices are not available in leaded packages, as are the more complex Spartan and all devices of the newest generation.

With a look at the decommissioning of the tools for older FPGA generations, the focus fell on Spartan 6 where two devices match: XC6SLX4 and XC6SLX9 [9]. Both are available in 144-pin QFP packages. The XC6SLX6 fails to provide the required memory. The XC6SLX9 has enough I/O pins, logic resources and memory to handle four FE chips. Additionally it contains everything required for the clock management needed by the configuration. Thus, those devices were not required to be external anymore, which was the case on the old eBOC.

Only one I/O standard is not supported: The ROD requires the clock to comply with the PECL (Positive Emitter-Coupled Logic) I/O standard[5]. This however can be compensated with an external clock buffer.

These features are supported with only two separate voltages, a 1.2 V core voltage and a 3.3 V voltage for I/O. This is acceptable as it is common for FPGAs with comparable features and does not introduce notable routing complexity.

[5]PECL uses emitter-coupled logic, which has a high switching speed at the cost of high current consumption. Due to this, it is especially suited for clock transmission.

3.2.2. The Printed Circuit Board

After defining the design constraints in the previous and this chapter, it is possible to map the configuration to physical components and create a Printed Circuit Board (PCB).

As mentioned in Section 3.2.1, the chosen FPGA is sufficient to support four FE-I4 chips and is capable of nearly all required I/O standards. Thus the ICs necessary to build the eBOC are two XC6SLX9 FPGAs with appropriate flash memories to store their configuration, a PECL driver for the clock to the ROD, a 40 MHz oscillator and a clock buffer to distribute this clock signal to the two FPGAs. Additionally a voltage regulator is required to supply the core voltage of 1.2 V of the FPGA. A block schematic of this mapping is depicted in Figure 3.10.

Each FPGA contains the logic to handle four FE chips and the complete logic to perform the required clock synthesis and configuration management. There are two differences between the two FPGAs: First, the North FPGA's `rodclk` output is discarded, whereas the South FPGA's `rodclk` output is fed into a PECL driver and then routed to the ROD. Second, the oscillator is fed into opposite I/O banks[6] of the FPGAs to simplify the routing of the clock.

The two separated clock syntheses are not a problem as all clocks are derived from the oscillator which is common for both FPGAs. Also, the DCMs and PLLs have a fixed phase-relationship between the input and the output. Thus, with the exception of minor differences due to different routing delays, the clocks synthesized on both FPGAs are equal in phase and frequency.

Figure 3.11 shows a front view on the layout of the PCB. The North and South FPGA are clearly visible in the center. North of each FPGA, DIP switches, LEDs and some I/O pins are available for debugging. The Setup Bus is connected to the northwest of both FPGAs and leads to the 160-way connector in the northwest corner of the PCB which is used to connect to the ROD.

In the southwest of the FPGAs, the Flash ROMs are placed which configure the FPGA. All four are connected to one JTAG bus, accessible through a $7 \cdot 2$-pin header.

At the south I/O banks, the command and data lines from and to the ROD are connected. The latter lead together with the ROD clock to the southern 160-way connector which leads to the ROD, too. The PECL driver for the ROD clock is set directly adjacent to this connector. Additionally, 5 V and 3.3 V are supplied through this connector.

The FE chip connectors are placed on the opposite side. Each has its own power connector to supply the FE chip directly at the eBOC. Of course, each FE chip connector

[6] An I/O bank is a a set of I/O pins which are supplied with the same voltages. In our case, the I/O banks are, with few exceptions, assigned to the four sides of the IC.

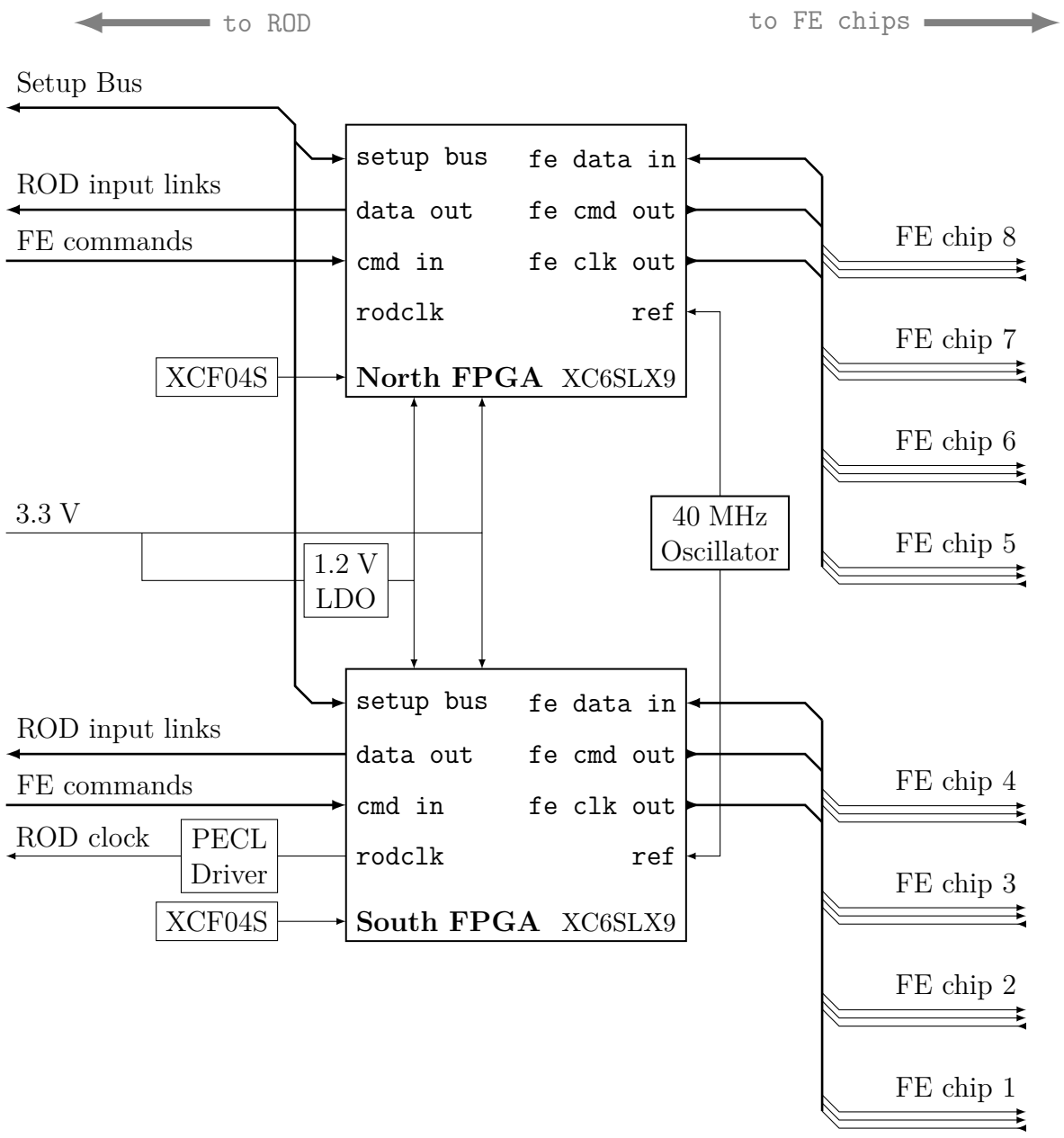

Figure 3.10.: Block diagram of the configuration mapped to the selected FPGA and extended by the necessary periphery.

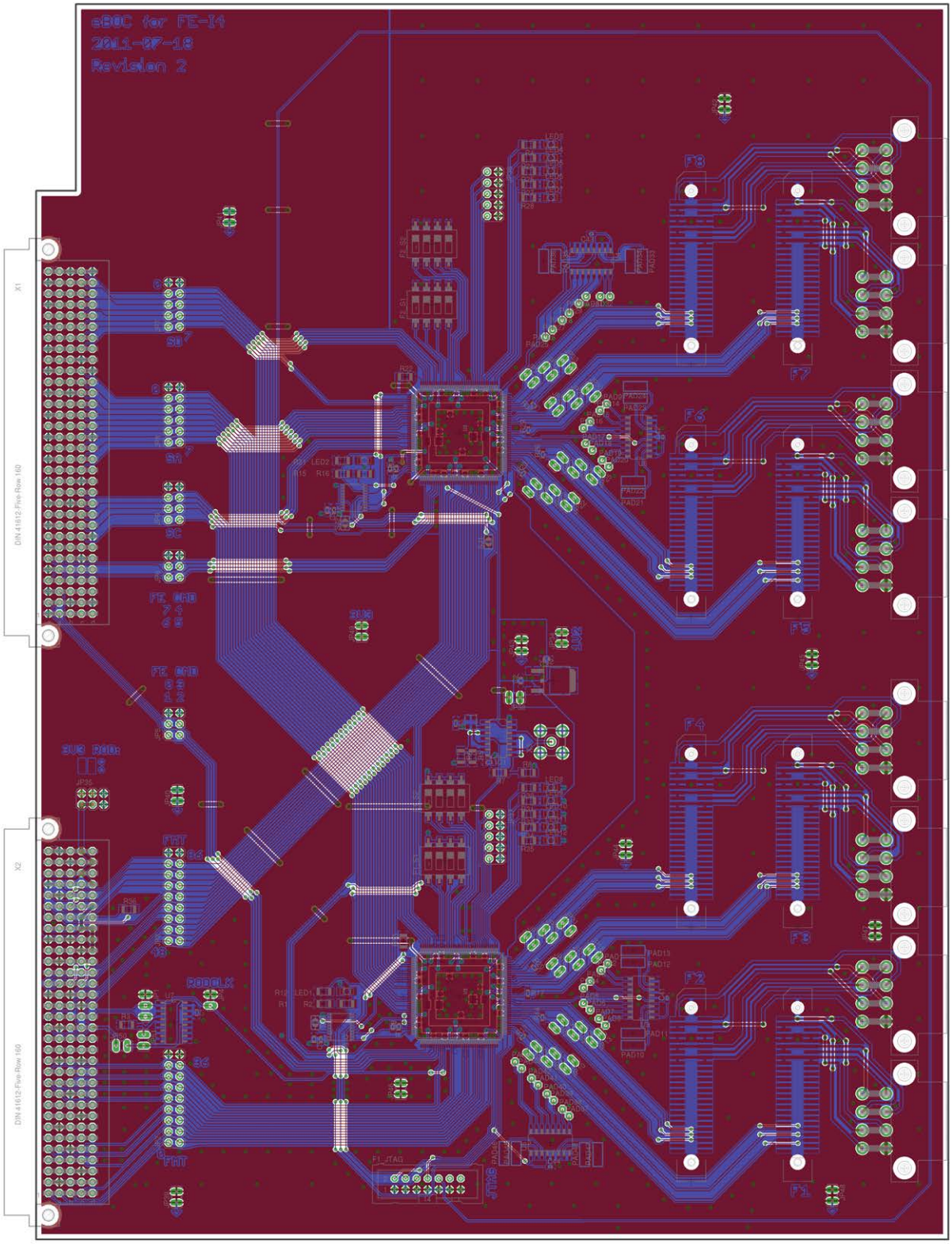

Figure 3.11.: Front view of the PCB design.

has a data, clock and command differential pair to the east side of one FPGA[7].

In the middle of the board, the 1.2 V voltage regulator and the oscillator with its buffer are placed. The clock buffer is also able to accept an external clock which can be supplied via an optional SMA socket[8].

All power rails, the aforementioned 5 V and 3.3 V which are supplied by the ROD and the 1.2 V rail are able to be manually switched on and off by removing a jumper. For debugging, the eBOC can also be powered from an external 3.3 V and/or 5 V power supply. The 1.2 V are always derived from the 3.3 V and cannot be supplied separately without removing the voltage regulator.

3.3. Changes to the MCC Emulator

The MCC Emulator was only able to transmit data at 40 Mbit/s prior to this thesis, but 80 Mbit/s and 160 Mbit/s were required for testing. Thus the MCC Emulator had to be reconfigured to comply to these parameters.

A simplified block diagram of the MCC Emulator is shown in Figure 3.12. Its main parts are the command decoder which is responsible for matching the input to the implemented commands, the Pixel Simulator which is controlled by a set of control lines and two separately wired registers. It outputs raw FE-I3 data which is then 8B/10B framed and encoded by the frame builder and encoder.

Those three blocks are supplied by a single DCM whose configuration allows to derive a 5 MHz and 4 MHz clock from the 40 MHz clock from the eBOC. Those two clocks are used solely by the frame builder. The 40 MHz clock from the DCM is fed into all three modules.

Due to the small time frame available for this thesis, a nearly completely external solution was chosen to allow higher bandwidths. This solution is depicted in Figure 3.13 and contains an additional DCM and a 1-bit-FIFO in the command path to allow stable handover of data from the input clock domain to the faster clock. The DCM increases the frequency of the whole Emulator by a factor r and thus, given successful timing analysis, has no impact on the logic itself.

The sole internal change has been done in the command decoder: It has a clock enable signal which is tied to the delayed and negated empty signal of the FIFO. Also, the FIFO is configured to automatically perform a read operation once it is not empty. Thus the

[7] In the current version, the eBOC layout still foresees some components which should not be populated, near the area that was just described. Those were planned to be used in case the FPGAs are unable to provide a signal acceptable to the FE-I4.

[8] A connector for coaxial transmission lines.

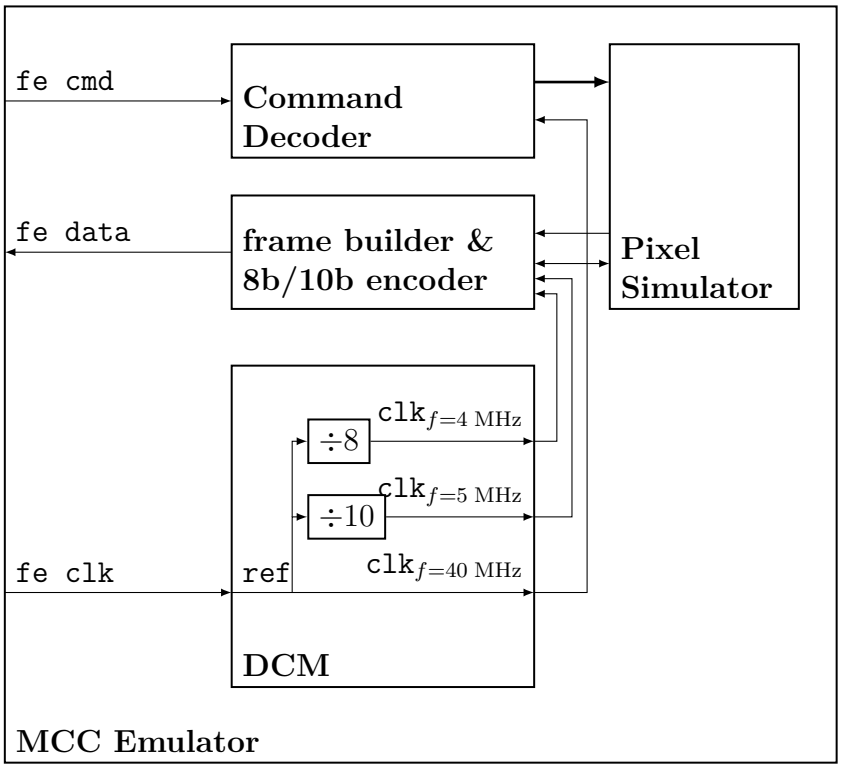

Figure 3.12.: Simplified schematic of the MCC Emulator as provided by Benjamin von Ardenne [7].

command decoder decodes only the next bit of the command line, once a new one is available. Also, this happens synchronous to the faster clock.

3.3.1. The Pattern Generator

The Pattern Generator is an alternative configuration for the MCC Emulator hardware which allows to send custom frames to the eBOC.

The configuration is relatively simple. As the MCC, it contains a command decoder, which in contrast to the MCC is only able to decode triggers, and an 8B/10B encoder. While the command decoder is clocked with the input clock, the rest of the configuration uses the output of a DCM and thus generally r times the input clock of 40 MHz.

For each scenario, the pattern generator can be supplied with a set of comma symbols and raw data as well as the desired data rate. Once a trigger is received, the pattern generator starts to encode this stream and transmits it to the eBOC.

This allows to easily reproduce any problems with the eBOC configuration and, due to its simplicity, an error can be quickly related to the eBOC or the pattern generator, which is not the case with the MCC Emulator.

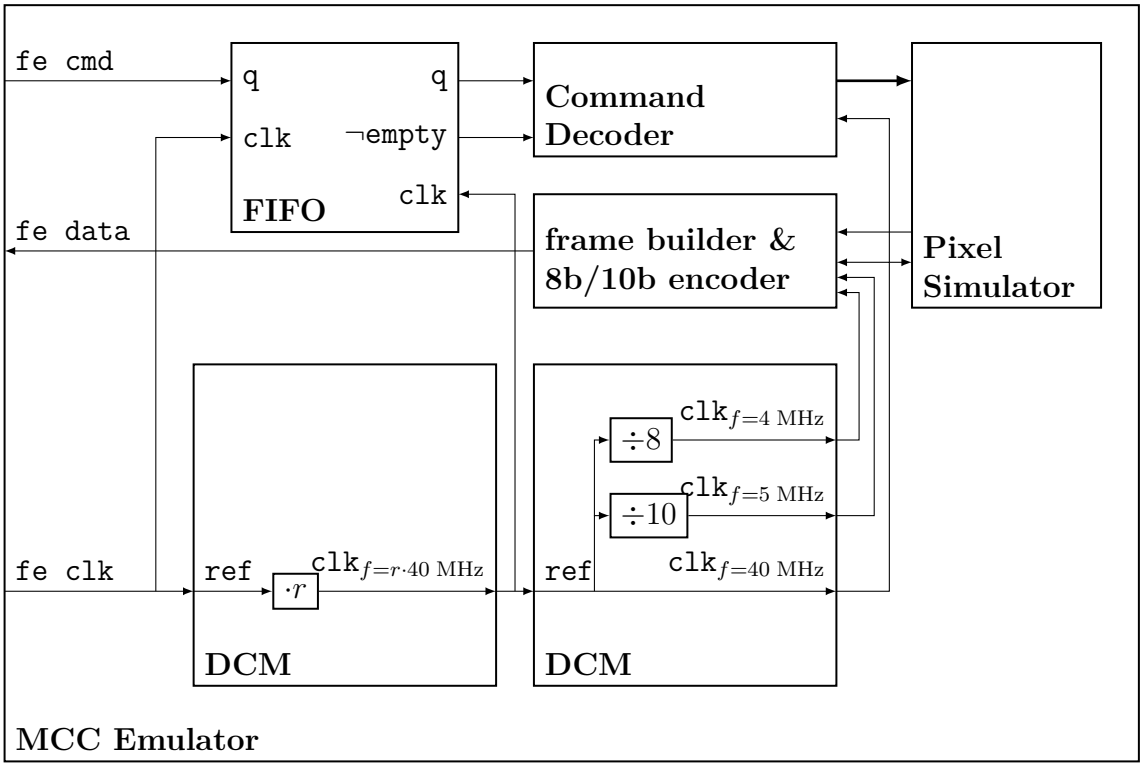

Figure 3.13.: Figure 3.12 extended by the modifications made to allow higher bandwidths.

4. Test and Validation

4.1. Simulation

The simulation performed concentrates on the receiver section due to the simplicity of the others. An 8B/10B encoder from Xilinx [23] has been put together with a serializer to create an input data stream for the receiver section. On the other side of the receiver section, the up to four channels are pushed into a shift register. Then the content of the shift register is compared with the input stream before the aforementioned 8B/10B encoder and asserts a signal once it has been detected.

There are three parameters that are accessible directly by this external simulation. The first parameter is the mode in which the buffer is working. Therefore, two test streams have been created: One is shorter than 1/5 of the maximum frame size to check whether the buffer operates if it has to buffer a full frame. The other is sufficiently long to check how the buffer performs if 1/5 of the maximum frame size is exceeded.

The second and third parameters affect the resynchronization circuitry: An initial phase shift φ_i between and a factor z on the clock frequencies of the receiver and sender. Though the latter should not occur with the FE-I4, this should be sufficient to simulate the phase shift of the clock caused by temperature changes.

The simulation has been performed with varying parameters, namely two frames with a size of 10 and 9994 Bytes, $\varphi = 0$ ns, 2.5 ns $\cdot$ z and $z = 99.5$ %, 99.9 %, 100 %, 100.1 %, 100.5 %.

4.1.1. Simulation Results for 40 Mbit/s, 10 Bytes, $\varphi = 0$ ns, $z = 1$

As an example, the simulation for 40 Mbit/s mode and small parameters will be described. Figure 4.1 shows the sync detector and data recovery behaviour for 10 Bytes, $\varphi = 0$ ns, $z = 1$. The sync detector has locked to the correct sampling point at 200 ns. Until 375.0 ns, the sync detector receives only non-8B/10B data, which is due to the delay of the sync detector and the intentional transmission of a short alternating sequence, which has been arbitrarily chosen to recreate the time before bootstrapping of the read-out system has been completed. Then the first synchronization symbol is received between

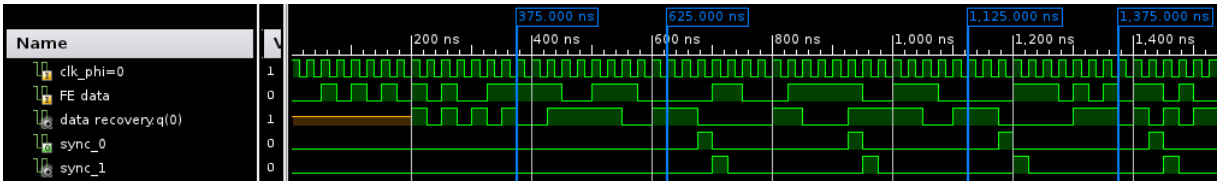

Figure 4.1.: Simulation of the sync detector and data recovery.

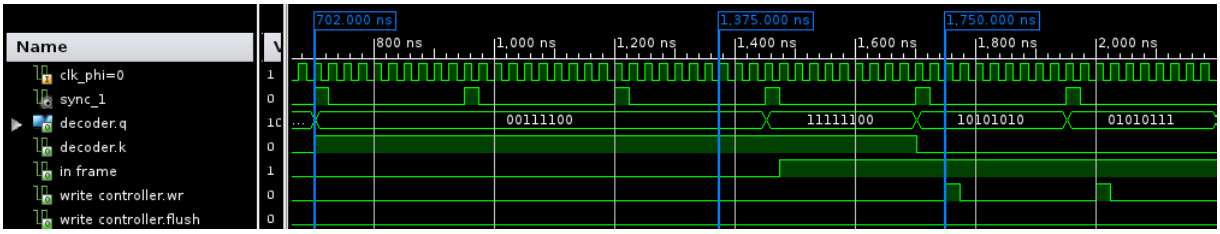

Figure 4.2.: Simulation of the write controller at the beginning of a frame.

375 ns and 625 ns (see Figure 2.2). Two clock cycles later the sync_0 signal is asserted. This delay is due to the delay of the deserializer and of the sync detector itself. This is followed by two additional ones until 1125 ns. Between this point and 1375 ns an SOF is received. sync_0 is asserted at the expected point of time.

Figure 4.2 shows the behaviour of the write controller, with the same absolute times as in Figure 4.1. Again, the reception of the three initial synchronization symbols is shown and they are decoded properly to their representation 00111100, k = 1. The same is the case for the SOF symbol. After the SOF symbol, the data is correctly latched into the FIFO using the *wr* signal, starting at 1750 ns.

The last data symbol has been decoded at 3950 ns with the EOF following 250 ns later. The write controller correctly detects the end of the frame and asserts the flush signal. The output controller picks up this signal and starts to read the first byte from the buffer

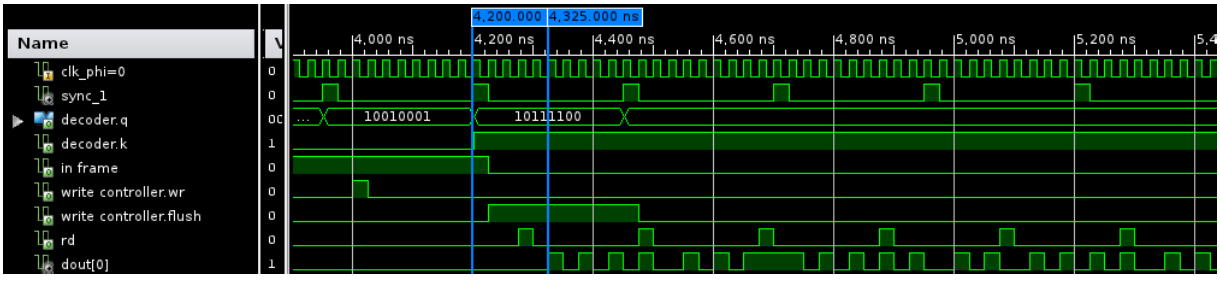

Figure 4.3.: Simulation of the write controller at the end of a frame and the interaction with the output controller.

FIFO at 4275 ns, the data is pushed into the output FIFO two clock cycles later. Though not shown in a Figure, the simulation continued until the whole data was successfully received by the simulation logic.

This, together with the successful simulations for the other parameters, shows that in general the logic devised for the eBOC is functional.

4.2. Timing Analysis

Timing analysis is automatically performed by the FPGA toolchain. However, the used timing analyzer is not capable to check live up to all constraints as it is unable to check timing across clock domains[1] [17].

The timing analysis yields that all constraints are met, except some in the data recovery module. These failures can be discarded as failsafe of the timing analyzer because of the aforementioned lack of functionality.

4.3. Measurements On A Prototype

A prototype of the eBOC PCB has been manufactured in-house. Due to some restrictions imposed by machining equipment which was brought into service only recently, differences between prototype and model exist. Those are mainly limited to additional copper areas with connection to ground or no connection at all. A photograph of the prototype is shown in Figure 4.4.

Two issues came up in this revision. First of all, a pin from the ROD was accidentally connected to the FPGA. As this pin was connected to the +12 V supply, this destroyed the FPGA. To replace the FPGA, two methods were considered. The FPGA could be mechanically cut out, but this would induce mechanical stress on the PCB pads. On the other hand, the FPGA could be desoldered with a professional desoldering tool[2]. The latter was chosen, which worked quite well. Only two pads, both leading to the debug connector, were destroyed during cleaning.

This solution is responsible for the second issue as it greatly added to the degradation of the through-hole plating. The resistance of all vias near the FPGA, where measurements could be performed, increased about one order of magnitude. One outliner even reached a resistance of 10 Ω. Most of the problematic vias were responsible for supplying the

[1] A clock domain is the range of the logic, where data is handled synchronously to a single clock. Once the data should be handled by a nother clock, this requires extra care to make sure the required timing constraints are met.

[2] Weller WQB 4000 SOPS

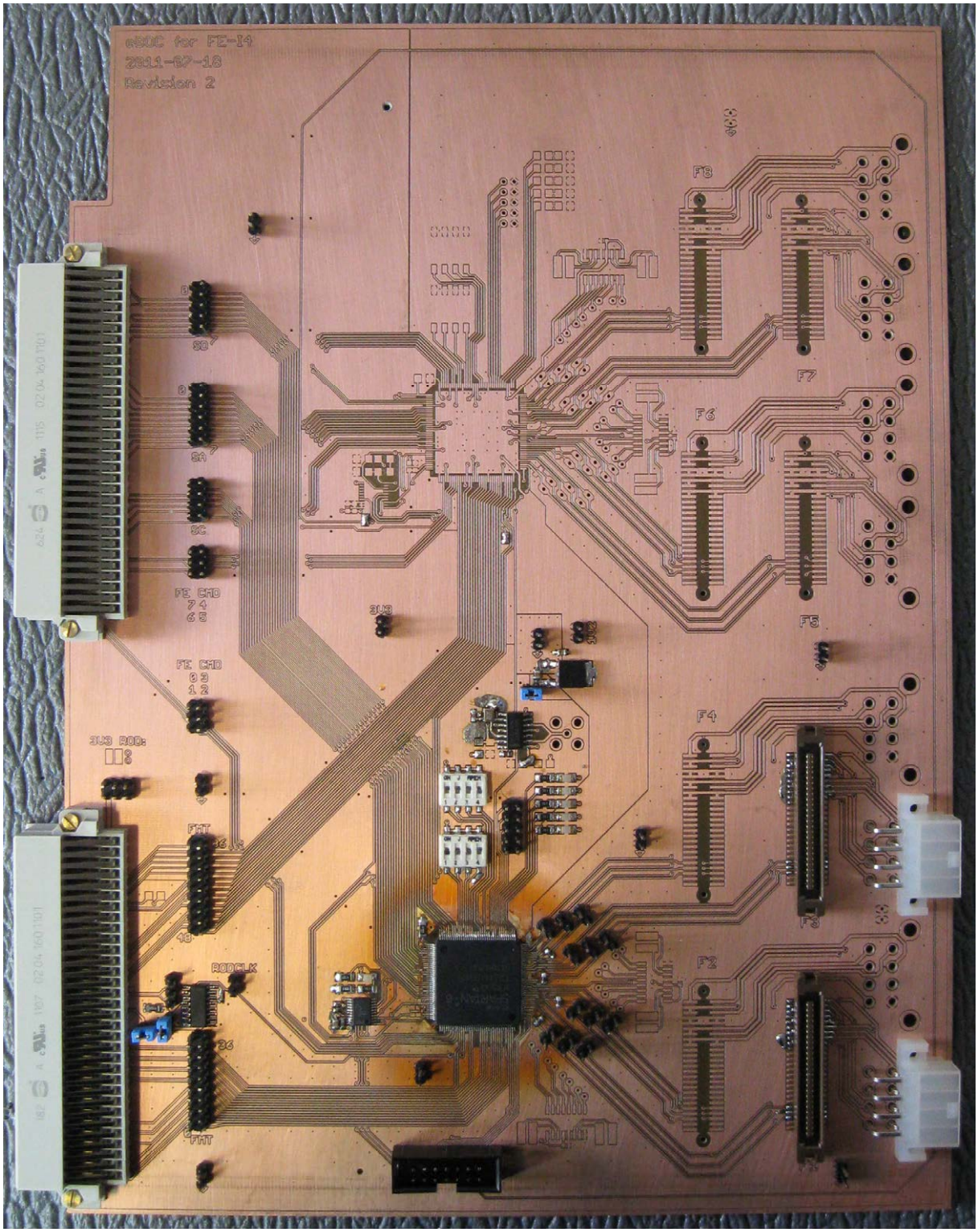

Figure 4.4.: The eBOC prototype.

FPGA's operating voltages, thus the performance of this eBOC is not optimal and may degrade further.

4.3.1. Clocks

To verify correct clock synthesis behaviour, the receiver clocks have been routed to the debug connectors. 0° and 90° phase shifts were observed in separate measurements, due to channel limitations on the measuring equipment and debug connector. As a reference, the `ref` input clock of the PLL that generates the output clocks, has also been routed to a debug pin.

Figure 4.5 shows the two measurements. In the top oscillogram, clocks with $\varphi = 0°$ are shown. By comparing the positions of the rising and falling edges, one can easily see that the clocks are aligned correctly. Also, the generated clocks' frequencies are 2^{r-1} of the reference frequency, as the waveforms were captured with persistence and thus they are stable. This observation also applies to the bottom capture. In addition, the 90° phase shift can be observed.

Also, the latching clocks generated by the DCMs have to be aligned to the data. Phase shift values of $\varphi_d = 0$, $\varphi_c = \varphi_b = 90°$ were configured. Figure 4.6 shows the ROD clock after the PECL driver and the data to the ROD. One can clearly see that the data is latched synchronous to the falling edge of the clock. Figure 4.7 shows the FE clock in reference to the commands read from the ROD. The same conditions are met.

4.3.2. LVDS Signal Integrity

The LVDS signal integrity has been measured at each LVDS receiver. Figure 4.8 shows the results: All signals have a voltage swing of about 600 mV or more, which greatly exceeds the required 100 mV [24].

4.3.3. Power Measurement

The power dissipation of this prototype has been measured to check if the values are within reasonable ranges and to contribute to the validation of future builds of this eBOC. The results are listed in Table 4.1.

During measurement, the eBOC has been kept within the air-cooled read-out crate, even if not connected to the ROD. The temperature, measured with a thermocouple directly behind the fan below the eBOC, stayed within 24.1 °C and 25.1 °C (±2 %).

Additionally, the expected power dissipation has been analyzed using the Xilinx XPower Analyzer. Of course, the analysis does not take into account the external oscillator and

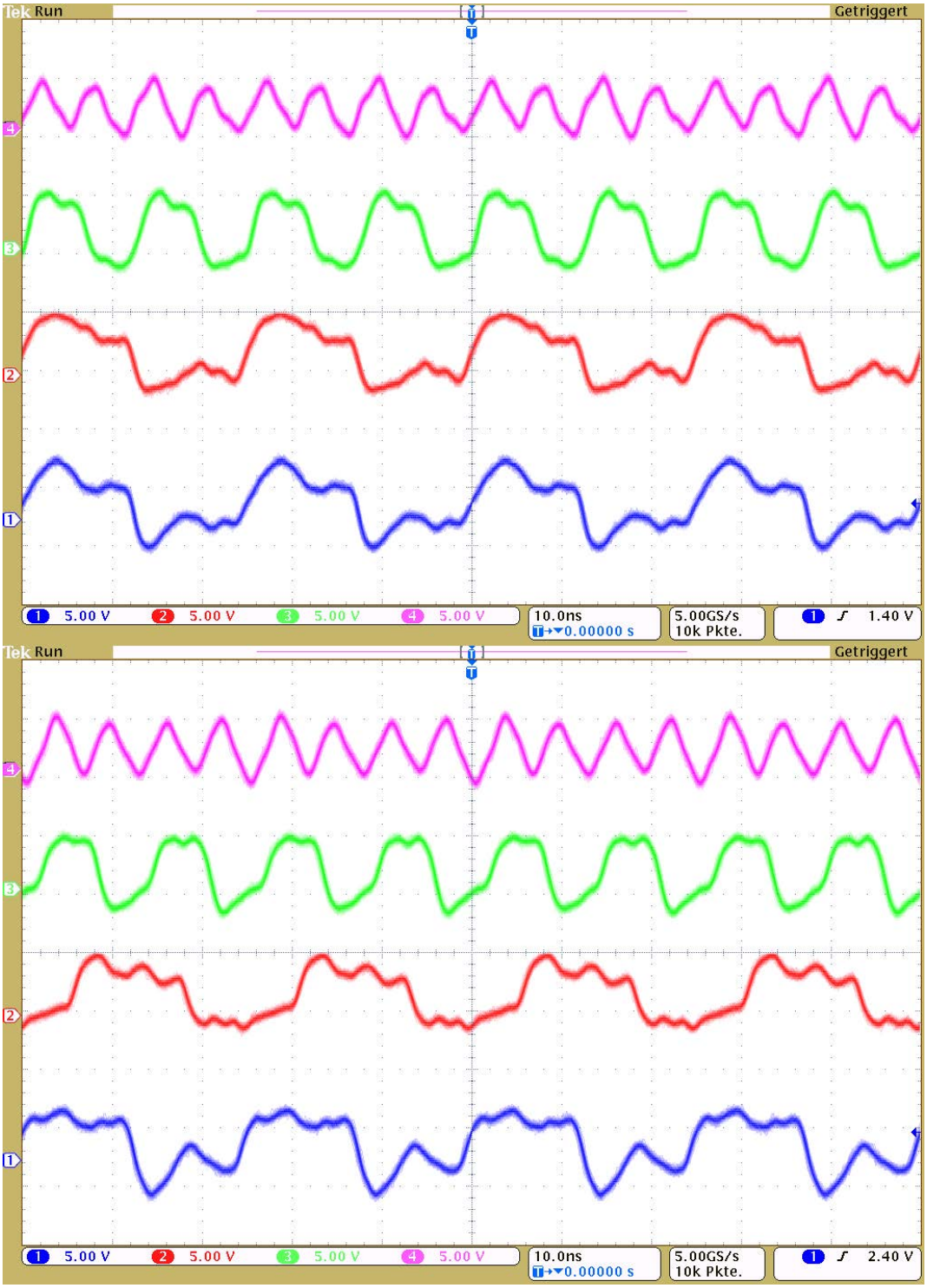

Figure 4.5.: Clock phase and frequency relationships around the PLL, measured at the debug connectors:
(top) 1. `ref`, 2. $\texttt{clk}^{\varphi=0°}_{f=40\text{ MHz}}$, 3. $\texttt{clk}^{\varphi=0°}_{f=80\text{ MHz}}$, 4. $\texttt{clk}^{\varphi=0°}_{f=160\text{ MHz}}$,
(bottom) 1. `ref`, 2. $\texttt{clk}^{\varphi=90°}_{f=40\text{ MHz}}$, 3. $\texttt{clk}^{\varphi=90°}_{f=80\text{ MHz}}$, 4. $\texttt{clk}^{\varphi=90°}_{f=160\text{ MHz}}$,
(see Figure 3.8 for a reference of the designators).

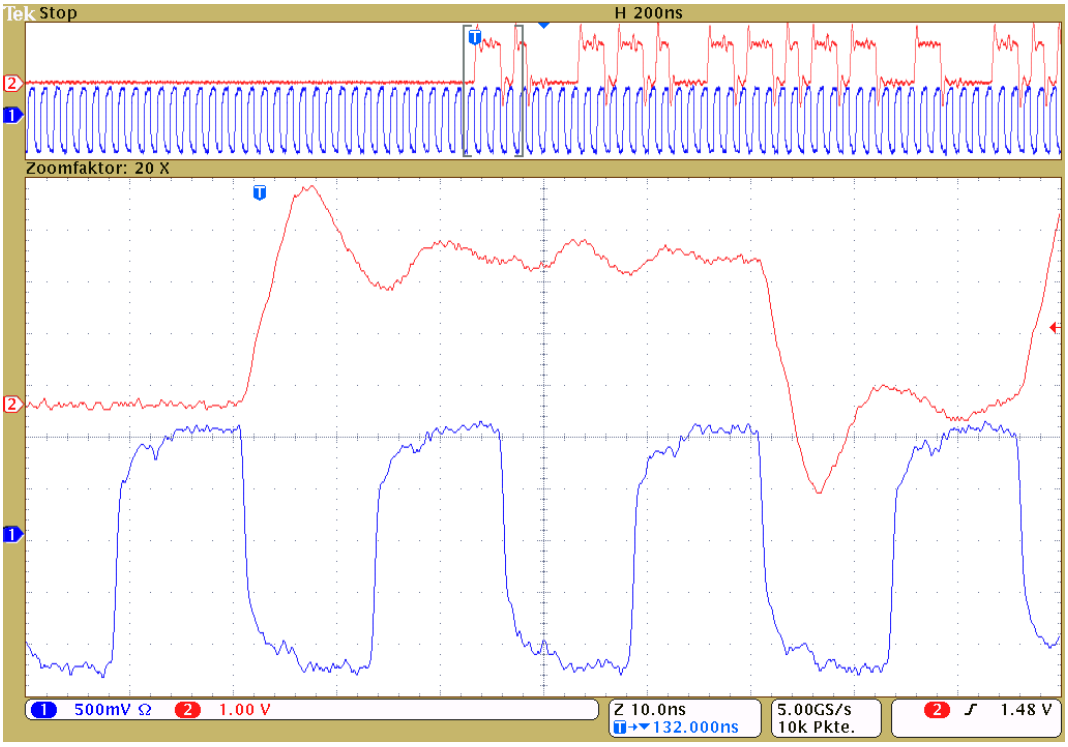

Figure 4.6.: ROD input data (2) in relation the the ROD clock (1).

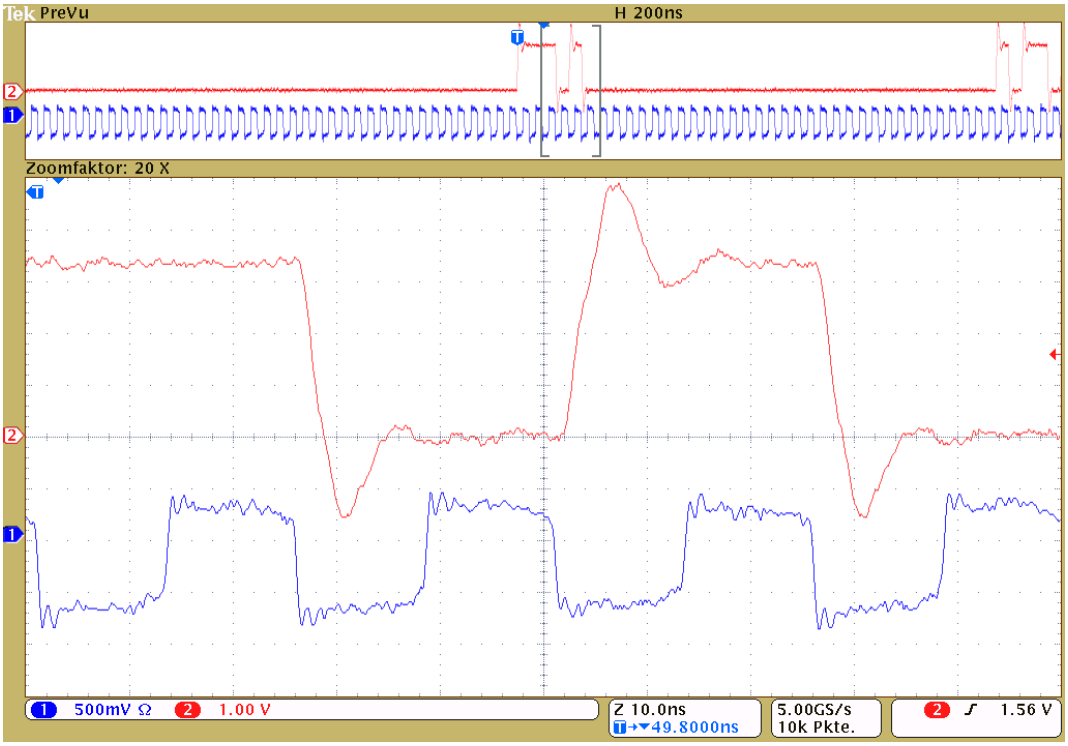

Figure 4.7.: Commands from the ROD (2) in relation to the FE clock (1).

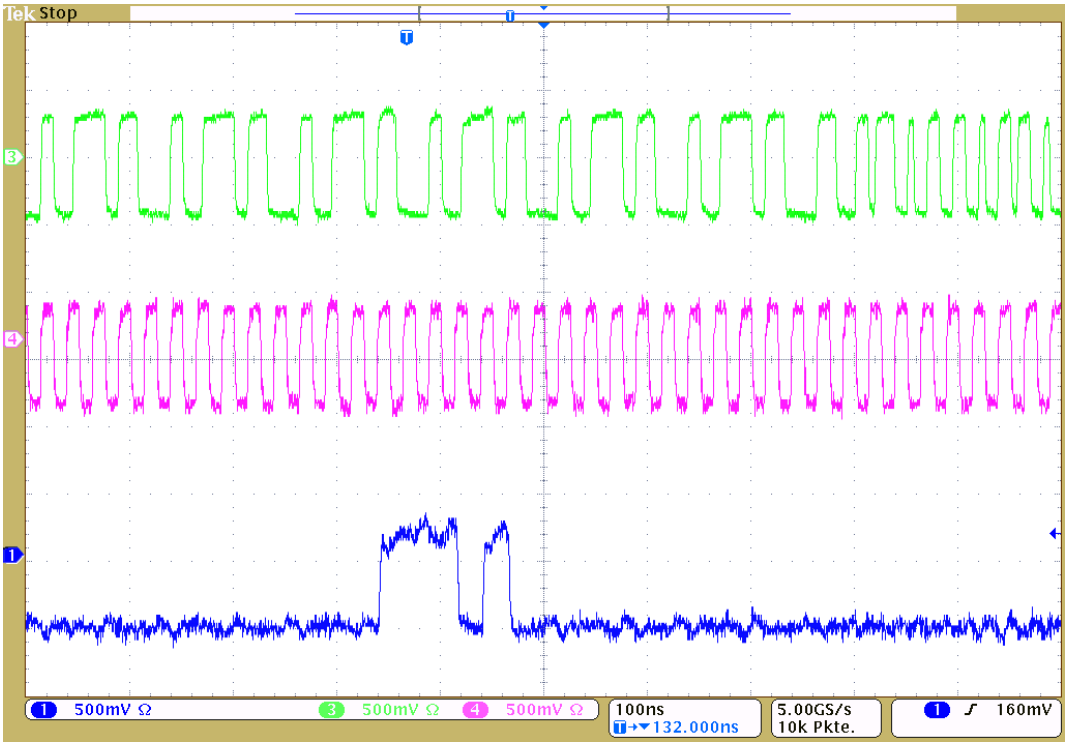

Figure 4.8.: LVDS signal integrity measurements, test point on receiver side. 1. FE command, 4. FE clock, 3. FE data

clock buffer, which alone require 22 mA, leaving a measured operating current of about 145 mA for the FPGA[3]. Comparing the results from Table 4.2 to the measured values, one can see that both match quite well.

4.4. Testing With The MCC Emulator

For functional validation, the eBOC has been connected to the ROD and one MCC Emulator, which for some tests has been programmed with the Pattern Generator configuration. On the ROD, a program called *MccRawViewer*, which was derived by Benjamin von Ardenne from a program written by Andreas Korn, is used. It initializes the ROD and sends triggers as they are requested by the user. To verify the result, it dumps the INPUT_MEM of the ROD and tries to decode the received data frame. The INPUT_MEM is the directly[4] after the inputs of the ROD and thus contains the data as it is processed by the ROD.

[3]This figure is based on the 160 MHz mode with all receivers enabled, nearly no current added for triggering as it is too small, but subtracting the measurement with the FPGA having no core voltage, as this is the current consumption of the oscillator and clock buffer.

[4]Actually, there is still a non-functional MUX in the way.

Parameters	Current (mA)
FPGA core voltage not supplied, eBOC not connected to ROD	22.2
All FPGA voltages supplied, eBOC not connected to ROD, FPGA not configured	30.5
All FPGA voltages supplied, FPGA not configured	33.1
40 MHz mode, no receiver enabled, no module connected	143
40 MHz mode, one receiver enabled, one module connected	143
40 MHz mode, one receiver enabled, triggering one module	144
80 MHz mode, no receiver enabled, no module connected	150
80 MHz mode, one receiver enabled, one module connected	151
80 MHz mode, one receiver enabled, triggering one module	151
160 MHz mode, no receiver enabled, no module connected	159
160 MHz mode, one receiver enabled, one module connected	159
160 MHz mode, one receiver enabled, triggering one module	161
160 MHz mode, all receivers enabled, no module connected	160

Table 4.1.: Current consumption of the eBOC in mA on the 3.3 V supply, ±2 % accuracy.

Pins	Voltage (V)	Current (mA)		
		Static	Dynamic	Total
VccINT	1.2	8	26	34
VccAUX	3.3	53	15	68
VccO	3.3	37	4	41
Total		98	45	143

Table 4.2.: Results of the power consumption analysis performed with the Xilinx XPower Analyzer.

For this thesis, the *MccRawViewer* has been modified to dump the correct ranges of the INPUT_MEM to compensate for the new link mapping. An error counter has been introduced to count errors produced by the decode routine. Also, the measurement can now be automatically repeated for long-term testing.

A major limitation of this approach is that the INPUT_MEM is set before the decoders which would multiplex the 40 Mbit/s input channels. To still allow testing of the eBOC with an intelligent decoder behind the ROD, the eBOC configuration has been slightly adapted for this test. Instead of mapping $r \cdot 40$ Mbit/s input data rate to r links to the ROD, only one link is used no matter what the input data rate was. This is possible due to the frame buffering on the eBOC, however, only frames that are actually smaller than $1/5$ the maximum frame size can be transmitted correctly. Frames that are considerably

Data Source	Data Rate ($\frac{\text{Mbit}}{\text{s}}$)	Duration (min)	8B/10B bits	8B/10B Errors	8B/10B Error Rate Upper Limit 90% CL	ROD Events	ROD Errors	ROD Error Rate Upper Limit 90% CL
PG	40	836	$2.0 \cdot 10^{12}$	0	$1.2 \cdot 10^{-12}$	$1.4 \cdot 10^{5}$	0	$1.6 \cdot 10^{-5}$
MCCE	80	1305	$6.3 \cdot 10^{12}$	0	$3.7 \cdot 10^{-13}$	$2.3 \cdot 10^{5}$	0	$1.0 \cdot 10^{-5}$
PG	160	806	$7.7 \cdot 10^{12}$	0	$7.2 \cdot 10^{-14}$	–	–	–
MCCE	160	2541	$24.4 \cdot 10^{12}$	0		$4.2 \cdot 10^{5}$	0	$5.5 \cdot 10^{-6}$

Table 4.3.: Results of the error rate measurements with the MCC Emulator (MCCE) and Pattern Generator (PG) and the upper limits for the error rates at the ROD and 8B/10B test points with 90% confidence level. In the PG/160 measurement, no event decoding was performed, thus there is no data available for event error rate measurement.

larger will create a buffer overflow in the eBOC as the FIFO read rate is less the write rate.

The MCC Emulator was additionally modified to support injection of channel inversion errors. This has been done to verify that the eBOC correctly detects errors in the 8B/10B coded transmission and the error counter of the *MccRawViewer* works.

4.4.1. Error Rates

For testing the bit error rate, the measurement has been automated to generate a large number of events. The error outputs of the decoder have been routed to a debug connector to which a storage oscilloscope has been connected. The oscilloscope was configured to trigger on rising edges of this output.

Before the beginning and after the end of each measurement, the data channel has been inverted to ensure correctness of the error detection setup. The values were then reset and the measurement ran several hours.

Table 4.3 lists the results of the error measurements. Given these values, an upper limit for the error rate with 90% confidence level have been calculated as already described in [6, 7] using

$$\varepsilon_{\text{err}} = 1 - \sqrt[n+1]{0.1}$$

with n being the number of events or bits, respectively.

Several errors that have been measured during these tests were not included in this table. This was done as the errors could be attributed to the attached programmer of the

MCC Emulator, as they only occured while other devices in the laboratory were switched on and off and the programmer was connected. Also, the MCC Emulator FPGA was always left unconfigured. This has already been observed by Matthias George [6], who came to the same conclusion.

The measured and calculated values show that the eBOC works stable over a reasonable time segment. The upper limits for the error rates for frames are not very low but largely depend on the time available for measurement.

4.4.2. Problems with the modified MCC Emulator at 40 Mbit/s

During testing, the modified MCC Emulator turned out to not work at 40 Mbit/s anymore. The problem could be tracked down to the frame builder and 8B/10B encoder block. As the data from the Pixel Simulator, which is available at a debug connector of the MCC Emulator, is still correct. This is particularly irritating, as the configuration in this area is still the same, it not simply operates at a higher frequency. Operation at this frequency however has been verified by the timing analyzer.

Due to time constraints and as the 40 Mbit/s mode is secondary, no further analysis has been performed on this error. However, it points to a currently undiscovered problem in the original MCC Emulator firmware that has not been discovered prior to this thesis and may cause problems if it is modified for other projects.

5. Summary

As a contribution to the development of the read-out chain for the new FE-I4 Pixel Detector chip for the IBL project, the electrical Back-Of-Crate card has been redesigned to fit the new requirements. Those were primarily the higher data rate per link and decoding of an 8B/10B data channel.

An FPGA configuration has been implemented which supports resynchronization of the data channel of the FE chip. Therefore, it allows plug-and-play handling of different cable lengths and tolerates other delay-changing parameters on the clock line. A vendor-supplied decoder has been used to handle the 8B/10B code, which eliminates most errors originating from a self made 8B/10B decoder, as this has already been tested thoroughly. Further parts of the logic are responsible for buffering and forwarding the data while demultiplexing it to up to four links to the ROD. With these capabilities in mind, a clocking scheme has been revised which reduces the logic required for the transmitter path and enables stable operation for all components through perfectly adjustable clocks.

This configuration has then been simulated with several varying parameters to validate its functionality. These parameters were chosen to recreate situations the eBOC is likely to encounter in operation.

Using the synthesis results of the configuration, a timing analysis and a set of other mostly physical parameters, an FPGA was chosen, a member of the Spartan-6 family, the XC6SLX9. A PCB has been designed around two devices of this type to support eight FE-I4 chips at 160 Mbit/s. The link mapping to the ROD has been corrected to that used by the original BOC. Additionally, the FE-I4 chips can be directly supplied through power connectors on the eBOC.

A prototype of this PCB has been created, hosting only one of the two FPGAs and connectors for two modules. It had to be assumed that the performance of this eBOC is degraded due to changes on the PCB imposed by the machining equipment and that parts of the PCB were exposed to heat beyond its specified range. After additional rework, an apparently functioning eBOC configuration could be programmed onto the FPGA.

The electrical performance of the prototype was tested. In particular, the operating currents of different functional units for their respective operation modes were measured.

Those numbers were compared to the result of a power dissipation analysis performed with vendor tools. Also, the performance of the differential transmission lines has been verified.

For testing, the MCC Emulator has been modified to support 80 Mbit/s and 160 Mbit/s using an external DCM which shifts the system frequency by a factor of 1, 2 or 4 and a modification which enabled decoding of the command input regardless of the system frequency. Also an alternative configuration for the MCC Emulator hardware, the Pattern Generator, has been programmed which was able to recreate the simulation performed of the receiver.

With the first test runs with the MCC Emulator, the clocks have been adjusted. Both adjustments have been verified by capturing the clocks in respect to their synchronous data channels.

At last, the setup was configured to perform a full test of the readout chain with the new eBOC and MCC Emulator. Errors in the 8B/10B and frame decoding have been recorded and none have been discovered. The test parameters have then been used to calculate an upper limit of the bit and event error rate.

6. Future Work

Now that an eBOC adapted to the FE-I4 has been prototyped and tested, the remaining parts of the read-out chain have to be ported. The new ROD is already on its way and thus the next step would be to build a new read-out chain with the new ROD, eBOC and a real FE-I4 module to port the ROD configuration to the new protocol.

Within the medium term, a new eBOC prototype should be built, due to the poor electrical performance measured. Through-hole plating of this type has already shown signs of further degradation over time in other projects and may cause problems and inconsistent behaviour of the eBOC. Also, in this revision, the connection from the FPGA to the pin BOC_AUX from the ROD should be removed. All components should be populated on this prototype to be able to perform a full validation.

Decoding the 8B/10B data on the eBOC introduces an additional buffer stage in the data path. This is generally not necessary and could be avoided by decoding the 8B/10B data on the ROD or accepting data at 32 Mbit/s instead of the fixed 40 Mbit/s per link.

Currently, the data rate and enabled modules of the eBOC are configured using DIP switches. This introduces an additional step which has to be taken into account when setting up the eBOC. It would be easier, if the eBOC could automatically detect connected modules and their data rates using the synchronization pattern that is transmitted on the silent data lines.

If the MCC Emulator should be used for other projects than this, the error 40 Mbit/s mode should be investigated as this points to an error in the configuration that may occur again in the future.

Currently, the only way for the eBOC to signal its status is using the status LEDs and the debug header, which is not feasible for debugging, which is a major task for a setup that contains this eBOC. Thus, the eBOC configuration should be extended to correctly work with the Setup Bus, as the BOC itself does. This could be used to dump the eBOC status or FIFOs, which is already possible at several points on the ROD.

A. 8B/10B Coding Map

5B/6B			
Name	Source	Encoded	
	EDCBA	abcdei	
		RD=-1	RD=+1
D.00	00000	100111	011000
D.01	00001	011101	100010
D.02	00010	101101	010010
D.03	00011	110001	
D.04	00100	110101	001010
D.05	00101	101001	
D.06	00110	011001	
D.07	00111	111000	000111
D.08	01000	111001	000110
D.09	01001	100101	
D.10	01010	010101	
D.11	01011	110100	
D.12	01100	001101	
D.13	01101	101100	
D.14	01110	011100	
D.15	01111	010111	101000
D.16	10000	011011	100100
D.17	10001	100011	
D.18	10010	010011	
D.19	10011	110010	
D.20	10100	001011	
D.21	10101	101010	
D.22	10110	011010	
D.23	10111	111010	000101
D.24	11000	110011	001100
D.25	11001	100110	
D.26	11010	010110	
D.27	11011	110110	001001
D.28	11100	001110	
K.28	11100	001111	110000

3B/4B			
Name	Source	Encoded	
	HGF	fghj	
		RD=-1	RD=+1
D.x.0	000	1011	0100
D.x.1	001	1001	
D.x.2	010	0101	
D.x.3	011	1100	0011
D.x.4	100	1101	0010
D.x.5	101	1010	
D.x.6	110	0110	
D.x.P7	111	1110	0001
D.x.A7	111	0111	1000
K.x.0	000	1011	0100
K.x.1	001	0110	1001
K.x.2	010	1010	0101
K.x.3	011	1100	0011
K.x.4	100	1101	0010
K.x.5	101	0101	1010
K.x.6	110	1001	0110
K.x.7	111	0111	1000

Bibliography

[1] O. S. Brüning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, P. Proudlock, *LHC Design Report*, CERN, Geneva (2004)

[2] A. Collaboration, *ATLAS detector and physics performance: Technical Design Report, 1*, Technical Design Report ATLAS, CERN, Geneva (1999)

[3] M. Capeans, G. Darbo, K. Einsweiller, M. Elsing, T. Flick, M. Garcia-Sciveres, C. Gemme, H. Pernegger, O. Rohne, R. Vuillermet, *ATLAS Insertable B-Layer Technical Design Report*, Technical Report CERN-LHCC-2010-013. ATLAS-TDR-019, CERN, Geneva (2010)

[4] M. Garcia-Sciveres, *ATLAS Experiment Pixel Detector Upgrades*, Technical Report ATL-UPGRADE-PROC-2011-006, CERN, Geneva (2011)

[5] F.-I. Collaboration, *The FE-I4A Integrated Circuit Guide* (2011), version 11.3

[6] M. George, *Implementation of an Electrical Read-Out System for Multi-Module Laboratory Tests of the ATLAS Pixel Detector* (2009)

[7] B. von Ardenne, *Development of new data transmission methods for the read-out system of the ATLAS Pixel Detector* (2010)

[8] Xilinx, *Virtex-6 FPGA Configurable Logic Block* (2009), v1.1

[9] Xilinx, *Spartan-6 Family Overview* (2011), v1.7

[10] Xilinx, *Spartan-6 FPGA Configurable Logic Block* (2010), v1.1

[11] Xilinx, *Spartan-6 FPGA Clocking Resources* (2011), v1.6

[12] Xilinx, *Spartan-6 FPGA Block RAM Resources* (2011), v1.5

[13] Xilinx, *Spartan-6 FPGA Memory Controller* (2010), v2.3

[14] P. J. Ashenden, *The Designer's Guide to VHDL*, third edition

[15] Xilinx, *Synthesis and Simulation User Guide* (2011), v 13.1

[16] Xilinx, *ISim User Guide* (2009), v 11.3

[17] Xilinx, *Timing Constrains User Guide* (2011), v 13.1

[18] Xilinx, *LogiCORE IP FIFO Generator* (2011), v8.1

[19] Xilinx, *Platform Flash In-System Programmable Configuration PROMs* (2010), v2.18

[20] A. Widmer, P. Franaszek, *A DC-balanced, partitioned-block, 8B/10B transmission code*, IBM Journal of research and development **27(5)**, 440 (1983)

[21] N. Sawyer, *Data Recovery*, Xilinx (2005), URL `http://www.xilinx.com/support/documentation/application_notes/xapp224.pdf`

[22] P. Vo, *Parameterizable 8b/10b Decoder*, Xilinx (2008), XAPP1112, v1.1

[23] P. Vo, *Parameterizable 8b/10b Encoder*, Xilinx (2008), XAPP1122, v1.1

[24] J. Brunetti, B. Von Herzen, *The LVDS I/O Standard*, Xilinx (1999), XAPP230, v1.1

Acknowledgements

I thank Prof. Dr. Arnulf Quadt and PD Dr. Jörn Große-Knetter for making this thesis possible, their outstanding organizational efforts and professional supervision, Nina Krieger for providing me with her knowledge about the setup, conceptual help and debugging help during commissioning of the prototype and support while writing this document, and the other members of the institute for it was a pleasure to work and spend time with them.

Thanks are going to the electronics workshop and especially Reinhard Mielke, who spared no efforts in providing me with a circuit board for the prototype, despite the long list of issues with the machining equipment.

Nonetheless I'd like to thank Sabine Wolff, my family and friends for supporting and bearing me for the last six months, which allowed me to write this thesis in the first place.

Erklärung nach §12(7) der Prüfungsordnung für den Bachelor-Studiengang Angewandte Informatik an der Universität Göttingen:

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe.

Göttingen, den September 29, 2011

(Johannes Agricola)