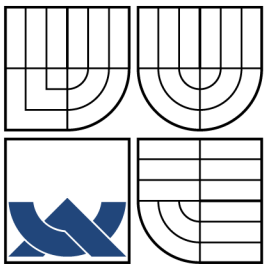
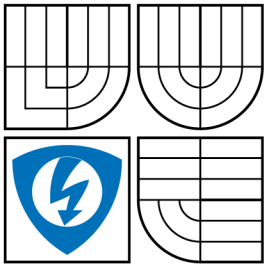


BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

# FAST BEAM CURRENT CHANGE MONITOR FOR THE LHC

MĚŘENÍ RYCHLÝCH PROUDOVÝCH ZMĚN ČÁSTICOVÉHO SVAZKU URYCHLOVAČE LHC

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE
JAN KRÁL

SUPERVISOR
VEDOUCÍ PRÁCE
Ing. DAVID BĚLOHRAD, Ph.D.

BRNO 2014

**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**Fakulta elektrotechniky a komunikačních technologií**

**Ústav radioelektroniky**

# Diplomová práce

magisterský navazující studijní obor
**Elektronika a sdělovací technika**

| | | | |
|---|---|---|---|
| ***Student:*** | Bc. Jan Král | ***ID:*** | 125500 |
| ***Ročník:*** | 2 | ***Akademický rok:*** | 2013/2014 |

**NÁZEV TÉMATU:**

## Měření rychlých proudových změn částicového svazku urychlovače LHC

**POKYNY PRO VYPRACOVÁNÍ:**

Seznamte se se systémem ochrany urychlovače LHC proti rychlým ztrátám částicového svazku (FBCCM). Navrhněte FBCCM a vhodnou metodu pro jeho komunikaci s řídicím systémem urychlovače LHC. Při návrhu zohledněte dostupné hardwarové prostředky, jednoduchost řešení a zavedené komunikační protokoly.

Realizujte navržené zařízení a jeho komunikační protokol s řídicím systémem urychlovače LHC. Navrhněte metodiku automatického měření zařízení v laboratoři a toto zařízení v praxi navrženou metodou otestujte.

**DOPORUČENÁ LITERATURA:**

[1] Cyclone III Device Handbook [online]. Vydáno: 2012, [cit. 5.6.2013]. Dostupné z: <http://www.altera.com/literature/hb/cyc3/cyclone3_handbook.pdf>

[2] Sitara AM335x ARM Cortex-A8 Microprocessors [online]. Vydáno: 2013, [cit. 5.6.2013]. Dostupné z: <http://www.ti.com/lit/ds/sprs717f/sprs717f.pdf>

| | | | |
|---|---|---|---|
| ***Termín zadání:*** | 10.2.2014 | ***Termín odevzdání:*** | 23.5.2014 |

***Vedoucí práce:*** Ing. Michal Kubíček, Ph.D.
***Konzultanti diplomové práce:***

**doc. Ing. Tomáš Kratochvíl, Ph.D.**
*Předseda oborové rady*

# ABSTRACT

Stringent demands on the LHC safety and protection systems require improved methods of detecting fast beam losses. The Fast Beam Current Transformer (FBCT) is a measurement instrument, providing information about bunch-to-bunch intensity of the accelerated beam. This thesis describes the development of a new protection system based on the FBCT signal measurements. This system, the Fast Beam Current Change Monitor (FBCCM), measures the FBCT signal in a narrow frequency band and computes time derivation of the beam signal magnitude. This derivation is proportional to the beam losses. When the losses exceed a certain level, the FBCCM requests a beam dump in order to protect the LHC. The LHC protection will be ensured by four FBCCMs which will be installed into the LHC in July 2014. Six FBCCMs have been already constructed and their characteristics were measured with satisfactory results. The FBCCMs were tested by a laboratory simulation of the real LHC environment.

# Keywords

Fast Beam Losses, Beam Intensity Measurement, FPGA, BeagleBone, DMA, Ethernet

# ABSTRAKT

Striktní nároky na systémy ochrany a zabezpečení LHC vyžadují vylepšené metody detekce rychlých ztrát částicového svazku. Rychlý proudový transformátor (FBCT) je měřicí přístroj poskytující informaci o intenzitě shluků urychlovaných částic. Tato diplomová práce popisuje vývoj nového systému ochrany LHC založeného na měření signálu FBCT. Tento systém, monitor rychlých proudových změn částicového svazku (FBCCM), měří signál FBCT v úzkém frekvenčním pásmu a počítá časovou derivaci okamžité amplitudy tohoto signálu. Tato derivace je přímo úměrná ztrátám částicového svazku. Pokud ztráty svazku překročí určitou úroveň, FBCCM přikáže kontrolním systémům zničit svazek kvůli bezpečnosti LHC. Ochrana LHC bude zabezpečena čtyřmi FBCCM, které budou instalovány do LHC v červenci 2014. Bylo zkonstruováno a testováno šest FBCCM. Jejich měřené charakteristiky vyhovují stanoveným požadavkům. FBCCM bylo vyzkoušeno laboratorní simulací reálného prostření LHC.

# Klíčová slova

Rychlé ztráty částicového svazku, měření intensity částicového svazku, FPGA, BeagleBone, DMA, Ethernet

# DECLARATION

I declare that I have made my Master's Thesis, with the subject of *Fast Beam Current Change Monitor for the LHC*, independently, under supervision of my supervisor, using the technical literature and the other information sources which are all cited in the text and enumerated in the list of the references at the end of this thesis.

In Brno, 23 May 2014

Jan Kral

# Bibliographic Citation

KRAL, J. *Fast Beam Current Change Monitor for the LHC.* Master's Thesis. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, 2014. 100 p. Supervisor David Belohrad

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor David Bělohrad for his invaluable technical advice and excellent management abilities. I could not have succeeded without his encouragement. The most difficult challenge for me was the initiation into the use of Emacs and the Linux OS. Today, I am glad that he convinced me to use these better methods. However, I could not have used these systems effectively without his advice and guidance. Last but not least, David, thank you very much for being a great friend.

My gratitude also belongs to my university supervisor Michal Kubíček for his administrative work and helpful advice concerning the study issues.

I would like to thank all my colleagues at CERN for creating a good working environment and for provided help. Especially I thank Josef Kopal and Pavel Fiala for taking the photo documentation of our project.


I would like to also thank for the offered opportunity to measure in the laboratories aided by the SIX project, the registration number CZ.1.05/2.1.00/03.0072, the operational program Research and Development for Innovation,

# CONTENTS

**Introduction** **3**

**1 Large Hadron Collider** **4**
1.1 Theory of Accelerator Operation . . . . . 4
1.2 Fast Beam Current Transformer . . . . . 6
1.3 Beam Synchronous Timing . . . . . 7
1.4 General Machine Timing . . . . . 7
1.5 Beam Interlock System . . . . . 7
1.5.1 User Interface of the BIS . . . . . 8
1.6 Fast Beam Current Change Monitor . . . . . 8
1.6.1 The First Prototype . . . . . 10
1.6.2 The Final Version of the FBCCM . . . . . 11

**2 FBCCM Hardware** **14**
2.1 ADC Subsystem . . . . . 14
2.1.1 Analogue to Digital Conversion . . . . . 15
2.1.2 RF Front End of ADC Subsystem . . . . . 15
2.1.3 ADC Clock Generation . . . . . 21
2.1.4 Clock Low-Pass Filter . . . . . 21
2.1.5 Single-Ended to Differential Clock Conversion . . . . . 24
2.1.6 Clock Termination . . . . . 24
2.1.7 TTC Receiver . . . . . 27
2.1.8 Indicators and Controls . . . . . 28
2.1.9 Serial Number . . . . . 30
2.2 CPU Subsystem . . . . . 30
2.2.1 BeagleBone . . . . . 31
2.2.2 Client Interface for CIBU . . . . . 32
2.2.3 Clock Input and Output, Telegram Receiver . . . . . 33
2.3 FBCCM Electric Power Distribution System . . . . . 35
2.3.1 ADC Auxiliary Power Supply . . . . . 36
2.3.2 Power Distribution of the ADC Subsystem . . . . . 37
2.3.3 Power Distribution of the CPU Subsystem . . . . . 38
2.4 PCB Design of the FBCCM Subsystems . . . . . 40
2.4.1 PCB of the ADC Subsystem . . . . . 40
2.4.2 PCB of the CPU Subsystem . . . . . 41
2.5 Mechanical Design . . . . . 41

**3 FPGA Firmware** **43**
3.1 Top Entity . . . . . 43
3.2 DIDT Master . . . . . 44

3.2.1 FIFO Buffer . . . 45
3.2.2 Moving Average . . . 46
3.2.3 Turn Flag Synchronisation . . . 54
3.3 BeagleBone Interface . . . 55
3.3.1 BeagleBone Initialisation . . . 55
3.3.2 BeagleBone Entity . . . 56
3.3.3 GPMC Slave . . . 57
3.4 Flash Data Upgrade . . . 60

**4 BeagleBone Software 61**
4.1 BeagleBone Bootloader . . . 61
4.2 BeagleBone Main Application . . . 62
4.2.1 DMA Module . . . 63
4.2.2 Main Loop . . . 65
4.2.3 FBCCM Server . . . 66
4.2.4 Structure of the FBCCM Messages Sent to Clients . . . 67

**5 FBCCM Measurements 70**
5.1 Measurements of the RF Front End Characteristics . . . 70
5.2 Laboratory Simulation of Losses . . . 72

**6 Conclusion 75**

**References 76**

**List of Abbreviations 78**

**List of Figures and Tables 80**

**Appendixes 83**
A.1 Schematic Diagrams of FBCCM Subsystems . . . 83
A.2 Photographs of FBCCM Subsystems . . . 87
A.2.1 Photographs of the ADC Subsystem . . . 87
A.2.2 Photographs of the CPU Subsystem . . . 89
A.2.3 Photographs of the ADC Auxiliary Power Supply . . . 91
A.3 Pictures of the FBCCM Mechanical Model . . . 93
A.4 Photographs of the FBCCM Device . . . 95
A.5 VHDL Codes . . . 98
A.5.1 FIFO Buffer Entity . . . 98
A.5.2 Flag Synchronisation Entity . . . 99

# INTRODUCTION

The Large Hadron Collider (LHC) is the largest and the most powerful particle accelerator in the world. LHC safety and protection is taken very seriously, especially after the damage to the LHC caused by a magnet quench in September 2008. After four years of operation, the LHC is currently in the Long Shutdown 1 (LS1) and many of its systems are being extensively upgraded. The main goal of the upgrade is to reach the nominal beam energy of 7 TeV instead of 3.5 TeV currently used. With circulating beams at twice energy, the LHC will be more vulnerable to the beam losses. Therefore the machine protection systems need to be absolutely reliable. The fast intensity measurement provided by the Fast Beam Current Transformers (FBCT) is used to measure fast beam loss and to implement a second level of beam loss protection in addition to the primary machine protection system based the Beam Loss Monitors [1].

This thesis focuses on the design, implementation and testing of the Fast Beam Current Change Monitor (FBCCM), fast beam loss detection system. The thesis is structured into the chapters. The first chapter briefly introduces the LHC operation, the FBCT intensity measurements and the other systems which the FBCCM is dependent on. This chapter also discusses the principal FBCCM operation, the first prototype of the FBCCM and the final concept of the FBCCM.

The principal FBCCM algorithm requires the digital processing of the FBCT signal. This processing is implemented in the FPGA of the Cyclone III Development Board. The analogue FBCT signal is converted for the digital processing by a dedicated electronic subsystem. The results of the signal processing are sent to the CERN Control Systems using the Ethernet network. The Ethernet interface is implemented by the microprocessor module BeagleBone. The BeagleBone is a part of another subsystem developed and described in this thesis.

The hardware development of the electronic subsystems is described in the second chapter. The third chapter looks into the implementation of the selected parts of the FPGA firmware. The BeagleBone software providing the FBCCM with the communication interface is examined in the fourth chapter.

Four FBCCMs will be installed in the LHC in July 2014 and be fully operational before the end of the Long Shutdown 1. Before installation, all the FBCCMs must be extensively tested in the laboratory to verify their functionality. Automatic testing methods were proposed and the FBCCMs were tested using these methods. The measurements and the testing are subjects of the fifth chapter.

The last chapter summarises the achievements of this thesis and presents following plans for the FBCCM project.

# 1 LARGE HADRON COLLIDER

Particle accelerators are devices generating high speed particles which are required in many diverse disciplines including fundamental and applied research, but also in industrial fields. Low-energy machines are for example cathode ray tubes frequently used in old screens or X-ray scanners in medicine and industry. Nuclei accelerated at lower energies are also used in the medicine as particle therapy, for treatment of cancer. Contrarily the high-energy particles are mainly provided for high-energy physics where experiments with them help nuclear physicists to explore new relations between elementary particles and extend knowledge of fundamental constituents of matter.

The Large Hadron Collider (LHC) belongs to the second category. Currently it is the most powerful and the largest collider in the world. It was built by the European Organization for Nuclear Research (CERN) and it is situated on the border between Switzerland and France close to Geneva. Its main purpose is to allow the physicists to test predicted theories of particle physics and based on results from experiments to prove or disprove these predictions. The monumental achievement of the LHC till now is the confirmation of the Higgs boson existence [1].

## 1.1 Theory of Accelerator Operation

Nowadays the particles in accelerators are accelerated by the electric field, whereas the magnetic field adjust the trajectory of the accelerated particles. To interact with the electromagnetic field, the accelerated particles have to be electrically charged, e.g. protons, electrons, and ions. The particles are accelerated in the vacuum, because otherwise they would collide with atoms of air and their acceleration would not be possible.

The trajectory of particles in the accelerator can be either circular or linear. Each type of trajectory has its advantages and disadvantages. The circular accelerators can benefit from the circular trajectory of the particle to accelerate particles each revolution which reduces demands on intensity of the accelerating electric field. The complication is that the particle path has to be curved by a magnetic field which the particles move in. The higher the velocity and mass of particles is, the stronger the magnetic field has to be to keep the on the defined trajectory. The linear accelerator basically requires higher intensity of the electric field to accelerate particles in the short time due to its limited size.

The particles are accelerated to speeds very close to the speed of light where laws of classical mechanics are not longer valid. The total energy of the accelerated particle due to Einstein's theory of relativity is given

$$E = \sqrt{(m_0 c^2)^2 + (pc)^2} \tag{1.1}$$

where $m_0$ is the invariant mass of the particle, $c$ is the speed of light, and $p$ is the momentum which is defined as

$$p = m_0 v. \tag{1.2}$$

The LHC is a circular accelerator with the circumference of 26659 m. Particles in the LHC are accelerated in two opposite directions and collide in places which are called experiments. The LHC accelerates particles from starting energy of 450 GeV up to collision energy of 7 TeV. The particles are accelerated by the RF electromagnetic field (respectively by its electric component) which is created in superconducting cavity resonators. The superconducting cavity is a part of the low level RF system of the LHC that is responsible for particle accelerating.

### Magnet Quench

The intensity of electric field in the cavities is 5 MV/m [2] and the magnetic induction of the bending magnets is up to 8 T [3]. This can be only achieved using superconducting cavities and magnets. Currently used materials become superconductive only at very low temperatures. The electromagnets in the LHC are cooled down to 1.8 K and temperature of cavities is about 4.5 K [3].

The superconducting electromagnets in the LHC are dipoles and quadrupoles. Their coils are made from superconducting wires that allow a current of 12 kA to flow to induce the nominal induction of the magnetic field. If the magnet temperature was increased by 1 K at nominal parameters, the magnet would stop being superconductive and become conductive with considerably higher resistance. The current of 12 kA flowing through the higher resistance would increase its temperature which would increase its resistance again resulting in the avalanche effect. The consequences would be very fast energy dissipation and very fast evaporation of the cooling liquid which could cause an explosion of the accelerator. This effect is known as a magnet quench.

### Beam Pattern in the LHC

The beam of particles in the LHC is not a constant flow. The particles are gathered in bunches separated by spaces. The particles in the bunch have a normal longitudinal distribution with a Gaussian probability density function. The number of particles in one bunch is called the bunch intensity. By summing bunch intensities over the revolution period (the entire turn), the beam intensity is obtained. In the LHC, one turn contains 3564 bunch slots, but because of technical reasons only 2808 bunch slots is usable. The bunch space between two consecutive bunch slots is the bunch period. It is approximately 25 ns and it is slightly reduced when the beam is being accelerated.

The bunch slots in the LHC are occupied by bunches based on the technical possibilities of preceding accelerators which inject the beam into the LHC and on the injection system. The occupied and free bunch slots form the beam pattern. The 2808 irregularly occupied bunch slots are followed by 3 µs gap which allows the extraction kicker magnet to turn on and reach the nominal intensity of the magnetic field. The extraction kicker magnet is a fast pulse magnet which ejects the beam

from its standard circular trajectory into the beam dump area where the beam is dissipated into heat.

## 1.2 Fast Beam Current Transformer

The beam is a flow of charged particles. The flow of particles generates a beam current. With respect to Ampère's law the motion of charged particles induces a magnetic field around the beam. The flow of particles is not constant, thus the beam current and the magnetic field is time variant. To sense the magnetic field (subsequently the beam current) a transformer around the beam is used. The transformer providing bunch to bunch measurements in the LHC is called the Fast Beam Current Transformer (FBCT).

The signal provided by the FBCT is proportional to the beam current generated by the motion of charged particles in the accelerator. It is used to estimate number of charges in the bunch (the bunch intensity), since the given integral of the beam current $i(t)$

$$\int_R i(t)\mathrm{d}t = Q \tag{1.3}$$

defines the charge $Q$ of the bunch when the integration region $R$ is limited to the bunch or the charge $Q$ of the beam when the integration region $R$ is limited to the revolution period. The number of charged particles in the bunch (respectively in the beam) can be estimated using the equation

$$Q = N \cdot e \tag{1.4}$$

where $N$ is the number of charges and $e$ is the charge of a proton.

The spectrum of the beam current is in the range from 0 Hz to ≈10 GHz. The bandwidth of the measured signal is limited by the FBCT and ranges from 300 Hz to 1.2 GHz. The spectrum of the signal corresponds with the beam pattern. The main component of the signal is at the frequency of 40 MHz which equals to the frequency of bunch slots $(25\,\mathrm{ns})^{-1}$. In addition many adjacent spurs spaced by 11.5 kHz are present due to the LHC revolution frequency.

There are four FBCTs installed in the LHC Point 4. Two FBCTs are installed in the system A and two in the system B. The system A is the system which actively runs the LHC. The system B is a spare system being used for development and testing. The one FBCT in the system A observes the beam 1, the other one the beam 2. The other FBCTs in the system B are organised in the same way.

The detailed information about intensity measurements and the FBCTs can be found in [4] or in the condensed version in [5].

## 1.3 Beam Synchronous Timing

The Timing, Trigger and Control (TTC) system distributes the Beam Synchronous Timing (BST) signal with trigger and control information around the machine [6].

The BST originates in the low level RF systems of the machine where the BST is derived from the RF frequency driving the accelerating cavities. High power lasers transmit the signal from the RF systems to the CERN Control Centre (CCC). In the CCC a signal carrying additional messages is generated synchronously with the BST. The result is a composite signal which allows a receiver to recover the 40 MHz bunch clock, the orbit turn flag, and variant BST messages. This composite signal is transmitted over single-mode optical fibres to the LHC experiments and to all beam instrumentation placed around the LHC ring.

The TTC provides instruments with the beam synchronous timing and messages. There is a lot of types of messages which can be transmitted over the TTC network, for example the beam momentum and the UTC time.

Further details can be found in [7].

## 1.4 General Machine Timing

The General Machine Timing (GMT) is a complementary system to the BST. The GMT uses the RS-485 physical layer. The GMT signal is broadcast at rate of 500 kbps [8]. The received timing signal has a jitter of 14 ns that is caused by the signal distribution over the copper cables and by the capacitive loading of the receiver inputs.

Besides the timing signal, the GMT carries a lot of messages sent over the network, e.g. the beam energy (equivalent to the beam momentum) and the UTC time.

## 1.5 Beam Interlock System

The Beam Interlock System (BIS) [9] is a part of the machine protection system. Its main goal is to collect information from the other systems that are fundamental in machine safety and decide if it is safe to inject the beam into the machine. When the beam is already being accelerated and any connected system fails or stops the machine operation, the BIS has to dump the beam.

The beam dump is performed by instructing kicker magnets to extract the beam from its natural trajectory and guide it into the beam dump where it is transformed into the heat.

More information can be found in [10].

### 1.5.1 User Interface of the BIS

The BIS provides all client systems with a User Interface (CIBU) to ensure a reliable machine protection. Each client system has to declare a safety flag permitting to inject the beam. Control systems permit the beam injection or request the beam dump depending on the status of the flags published by the client systems.

The most important factor of the BIS is the reliability; therefore the CIBU is equipped with two separate channels. When the device sets the beam permit flag, it has to set both channels to their active states independently. This duplicity minimises a possibility of setting the permit flag by mistake due to a failure of output circuits.

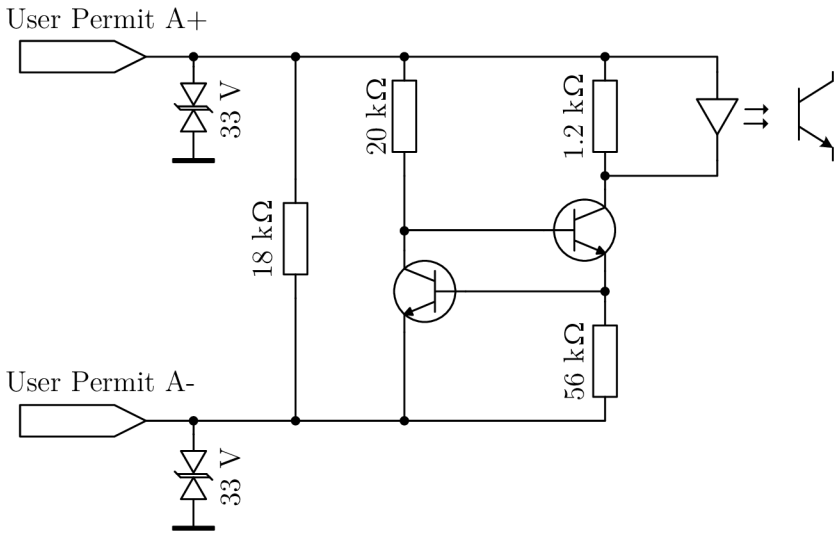



Fig. 1.1: CIBU input circuitry

Fig. 1.1 depicts the input circuitry of the one of the CIBU channels as it is defined in [11]. Referring to the document of the CIBU [11], the channel is considered in the active state when a current of more than 9 mA flows from the port A+ (respectively B+) to the port A- (respectively B-) of a connected system. The beam permit flag is set when both CIBU channels (A and B) are set to their active states.

Twisted pair cables with two Burndy 8-pin connectors are used to interconnect the CIBU and the client system [11].

## 1.6 Fast Beam Current Change Monitor

The Fast Beam Current Change Monitor (FBCCM also known as DIDT) is a beam diagnostic instrument for the LHC. The FBCCM is a redundant system of the machine protection to the LHC Beam Loss Monitors (BLM) [12]. The Beam Loss Monitors are devices which protect the LHC against damages caused by failures of the machine systems. They are inserted into the beam path and integrate the

current through the ionisation chamber to estimate the number of particles in the beam. When the BLM detects fast losses, it triggers the beam dump to protect the machine.

The FBCCM principle is different and independent of the operation of the Beam Loss Monitors. It evaluates the measurements of the beam intensity provided by the FBCT and calculates losses.

The motivation for the FBCCM is to increase the reliability of the machine protection provided by using a different principle of measurements. Additionally the FBCCM has a potential to achieve faster detection of the losses than the BLM.

The idea of the FBCCM is to acquire the signal from the FBCT and to evaluate the losses of beam intensity. In the accelerators no losses of intensity are ideally acceptable, but because of imperfections of the machine some losses are tolerated. When the FBCCM detects the fast losses exceeding a tolerated level, it has to trigger the beam dump.

Experimentally it was discovered that the beam intensity is proportional to the magnitude of the FBCT signal in the narrow band around the 40 MHz component. This property is used by the FBCCM to estimate the beam intensity. The magnitude of the 40 MHz component is integrated to calculate the total number of charges over different time (integration window). Ideally the result of the integration over the each window should never be less than the previous result of the same window. The negative differential of definite integral would signify losses of the charges. The condition for non-zero losses would be

$$\frac{\mathrm{d}}{\mathrm{d}\tau} \int_{W} |s(t)| \mathrm{d}t < 0 \tag{1.5}$$

where $|s(t)|$ is the magnitude of the signal proportional to the beam intensity and window $W$ is the region of the finite integral. However in the real system some losses are tolerated, then the FBCCM has to dump the beam when the following condition met.

$$\frac{\mathrm{d}}{\mathrm{d}\tau} \int_{w} |s(t)| \, \mathrm{d}t < \vartheta_W \tag{1.6}$$

where $\vartheta_W$ is the tolerated loss threshold for the particular window and energy.

Based on the study of intensity measurements it was decided that the length of different integration windows for the FBCCM implementation should be 1, 4, 16, 64, 256, 1024 turns. To comply with all requirements of the narrow bandwidth and six different integration windows, the signal processing has to be performed digitally. Eq. 1.6 can be approximated into the discreet domain as

$$\sum_{y-N}^{y} |s_x| - \sum_{y-2N}^{y-N} |s_x| < \vartheta_N \tag{1.7}$$

where $N$ is the length of the summation window in the number of input samples of the signal $s_x$ and $\vartheta_N$ is the threshold of tolerated losses for the particular length of the window and energy.

### 1.6.1 The First Prototype

Belohrad presents the first prototype of the FBCCM in [13] and shows the system concept (Fig. 1.2). The analogue signal from the FBCT is filtered by the anti-aliasing low-pass filter (LPF) module to suppress the frequencies above the frequency of 40 MHz which the band of the interest is around. The filtered signal is converted into the digital domain by the ADC ADS5485 using the TI evaluation module [14]. The evaluation module is then connected through the High Speed Mezzanine Card (HSMC) connector to the Cyclone III Development Board. The digital processing is implemented in the FPGA.

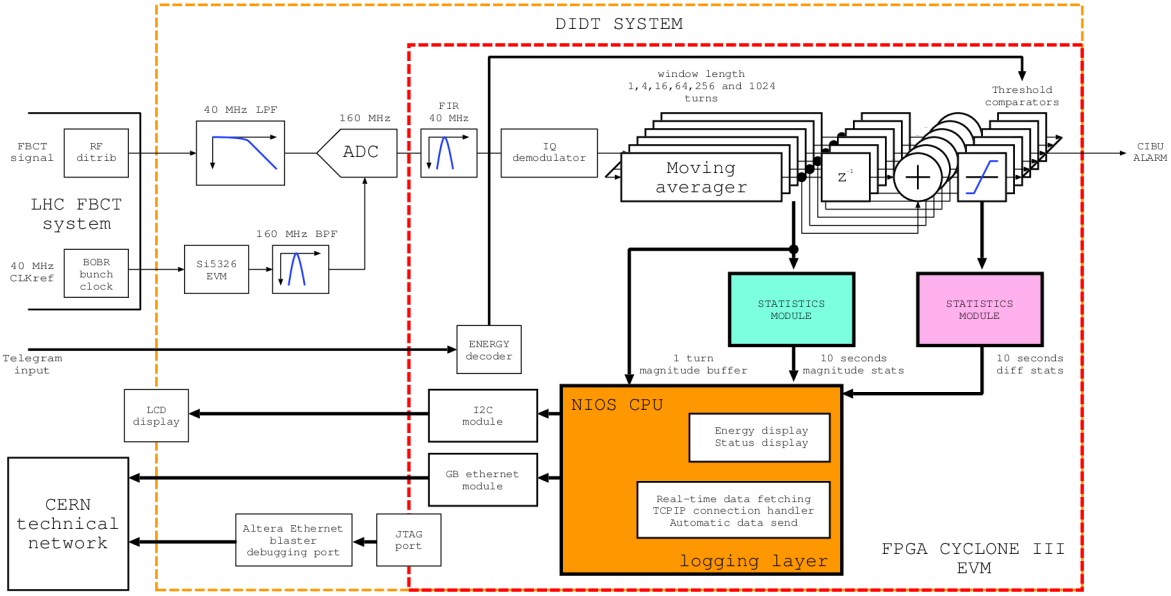



Fig. 1.2: Diagram of the FBCCM system [13]

The digital signal is firstly filtered by the narrow-band FIR filter with the bandwidth of 2.5 MHz. Referring to Eq. 1.7 magnitude of the signal has to be calculated. The digitally filtered signal is firstly translated by the quadrature mixer to the baseband around frequency of 0 Hz. The magnitude of the signal is then calculated from its $I$ and $Q$ samples.

$$|s_x| = \sqrt{I^2 + Q^2} \tag{1.8}$$

The simplest quadrature mixer for the translation of the high-frequency digital signal to the baseband can be implemented when the sampling frequency of the signal is four times the signal frequency. The samples of the signal are then equal to the sequence of samples $+I$, $+Q$, $-I$, and $-Q$. This implementation is only possible if the sampling clock is locked to the beam and its frequency is four times higher than the bunch frequency. To comply with this condition the bunch clock from an adjacent system is provided to the first prototype. The PLL Si5326 is used to multiply the bunch frequency to obtain the 160 MHz sampling clock. The ADC achieves the better signal-to-noise ratio when the sampling clock waveform is the

sine wave instead of the square wave. The generated phase-locked square wave is therefore filtered to suppress higher harmonics before it is connected to the ADC.

The signal processing continues by summing the magnitude signal over windows with different lengths and computes the difference of two consecutive output samples for each window. When the difference exceeds the defined threshold, the FBCCM triggers the beam dump. The thresholds are selected with respect to the actual beam energy which is decoded from the messages in the received telegram from the GMT.

The outputs of the moving average and the differences are statistically evaluated. The results (maximum, average, minimum) are sent each ten seconds by the NIOS processor to the CERN technical network over Ethernet The NIOS processor is an Altera's soft processor implemented into the general FPGA logic. The data is sent through TCP sockets to ensure the reliability of their reception. Any general client can connect to the FBCCM to receive the data. Besides these statistical messages the NIOS periodically sends data of the one turn moving average and as well the beam dump messages anytime the beam dump occurs. At the same time it prints device status information in the front panel LCD display.

The management and updates of the device are performed via the Altera Ethernet Blaster which is a JTAG programmer with the Ethernet connection. This programmer is fully supported by Altera and it allows to execute remotely the same operations like an ordinary USB JTAG programmer. The Altera Ethernet Blaster provides a secured way to remotely update and debug the device.

Generally the first prototype is very modular which is great for a testing and an elaboration, because each module can be replaced with another having different characteristics or functionality. Unfortunately the most modules were not specifically designed for the FBCCM, thus in some cases the system is over-complicated. These characteristics are not convenient for the final solution.

### 1.6.2 The Final Version of the FBCCM

The first prototype of the FBCCM was never intended to be a final solution. It was designed and constructed to prove the measurement methods. Some parts and features of the first prototype were known not to be ideal even before the first tests, another were discovered lately after the examination of the test results. The outcomes of the first prototype testing defined specifications for the final version of the FBCCM.

The first prototype was equiped with two measurement channels and measured intensity of both beams at the same time. This feature turned up as a disadvantage in the end, because a cross-talk between the two channels degraded the sensitivity of the instrument. Even though the cross-talk is a hardware issue and a proper design should minimise it to the sufficient level, it was decided to construct one device for each beam.

The soft processor NIOS of the previous system had to be replaced, because as

every soft processor, it consumes a lot of valuable high-speed FPGA logic which is needed for the signal processing. Moreover the TCP/IP stack implemented in its operating system $\mu$C/OS was not stable enough and the processor was too slow to handle the TCP communication with more than two connected clients. This performance limitation denies implementation of expert features in the future.

The first prototype acquires the beam synchronous timing and the telegram from two different interfaces. The final design combines the reception of both timing and the telegram into the common receiver of the TTC received from the optical fibre. This improves reliability of the FBCCM due to the less cabling and removes the FBCCM dependence on the adjacent system. Moreover it was discovered in the first prototype tests that settings performed on the adjacent system influenced the FBCCM functionality. Therefore the FBCCM requires a dedicated link for the timing reception.

This thesis is aimed at the development of the new device implementing fixes of all the discovered issues of the first prototype. Six operational pieces of the new FBCCM had to be fabricated and subsequently tested in the laboratory. In the end of the testing phase, four devices will be installed in the LHC close to the measuring FBCTs in the LHC Point 4, while two devices will be kept in the laboratory for testing and development purposes. Two devices will supervise the beam 1 and 2 and will be in the system A and the others for the beam 1 and 2 in the system B. The organisation of the FBCCM in the LHC is similar to the organisation of the FBCTs.

The desired concept of the new system is shown in Fig. 1.3. The measurement principles and algorithms are well tested in the first prototype; therefore they are adopted by the new FBCCM. Nevertheless their implementation is absolutely different. There was an intention to replace the NIOS processor with a system on chip in Cyclone V, but unfortunately a release of this FPGA was delayed and the development of the FBCCM had to meet the given schedule. Therefore the Cyclone III Development Board was kept in the design and a BeagleBone was chosen as a suitable replacement for the NIOS processor. The BeagleBone is a microprocessor system using the AM335x Sitara ARM Cortex-A8 processor. It is equipped with the Ethernet connection which is used to transport the measured data to the CERN Control System.

Two peripheral subsystems for the FBCCM had to be designed. The each subsystem is connected to the HSMC connector on the central FPGA board. The ADC Subsystem contains the analogue-to-digital converter, the main TTC receiver, indicators, and controls. The CPU Subsystem interconnects the BeagleBone with the FPGA board. Furthermore it carries the CIBU interface, the spare TTC receiver, the telegram receiver, indicators, and controls.

The power subsystem in the new FBCCM is completely different, because previously used evaluation and development boards required various voltages which are not more needed. This makes the power system simpler.

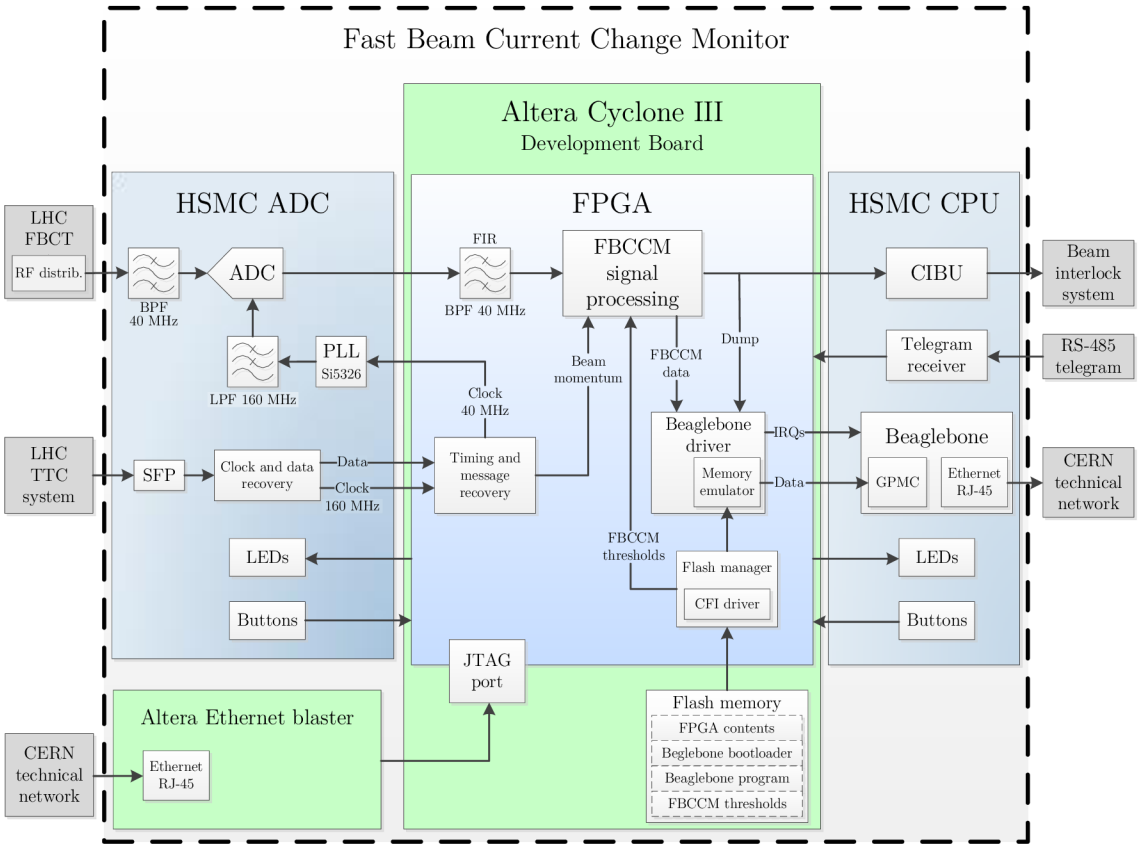



Fig. 1.3: The new FBCCM system concept

# 2 FBCCM HARDWARE

This chapter focuses on the design of the FBCCM hardware. The main designed electronics are the ADC Subsystem and the CPU Subsystem. The subsystem design is restricted by the possible signal connections to the central Cyclone III Development Board.

## 2.1 ADC Subsystem

The ADC Subsystem is one of two designed peripheral boards connected by the HSMC connector to the FPGA module. It contains a mixture of digital and analogue electronics. Its block diagram is shown in Fig. 2.1. The complete schematic diagram can be found in Appendix A.1.

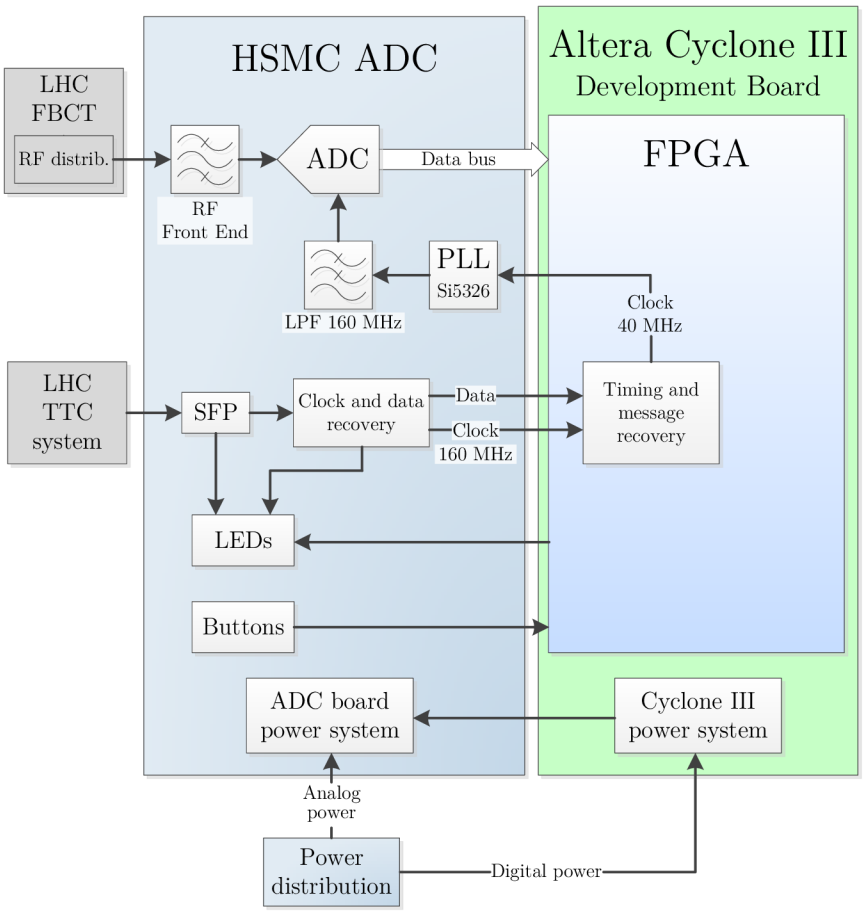



Fig. 2.1: Block diagram of the ADC Subsystem

The digital electronics include high speed data buses communicating with frequencies up to 160 MHz. These buses require a proper impedance matching to avoid signal reflections. The reflections would narrow the eye-diagram of the signal at the

destination and shorten the time window of the proper sampling. Furthermore the lengths of signal traces in common bus have to be matched to ensure equal propagation delays for all the signals in the bus. Different transport delays would shorten the sampling window for the whole bus. The narrow window of a proper sampling would put higher constraints on timing driven compilation of the FPGA.

The impedance matching is the issue of the analogue electronics design as well. The reflected signal would be superposed to desired signal and negatively influence its processing.

### 2.1.1 Analogue to Digital Conversion

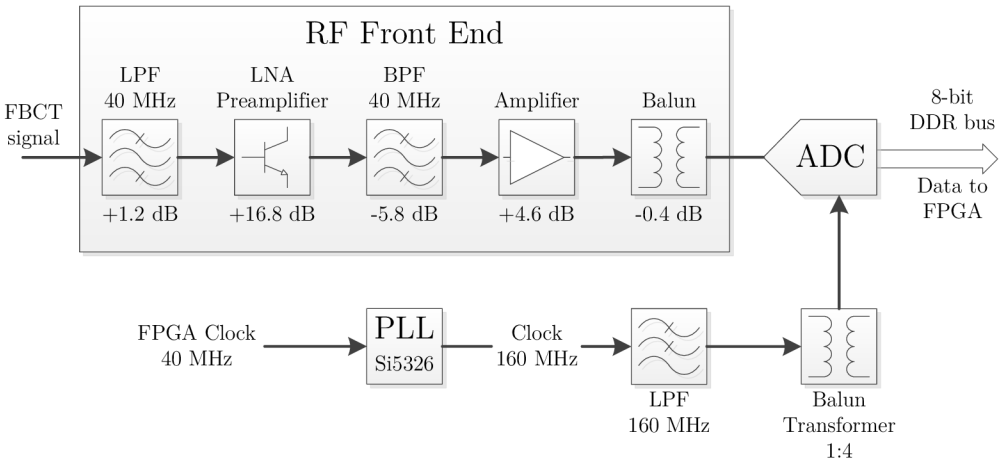



Fig. 2.2: Block diagram of the analogue to digital conversion of the ADC Subsystem

The main purpose of the ADC Subsystem is to convert the FBCT signal into the digital domain. The input signal level is too low to be converted directly. It has to be gained and filtered. The filtration suppresses strong frequency components out of the processed band. Otherwise these components would degrade the system signal-to-noise ratio (SNR), because the signal gain would be limited to the ADC full-scale range by these undesired components. The signal is gained and filtered by the RF Front End.

### 2.1.2 RF Front End of ADC Subsystem

The design and development of the RF Front End is not part of this thesis. Its structure and values of its components have been adopted from the first FBCCM prototype. The implemented circuit of the RF Front End is depicted in Fig. 2.3. The particular stages of the front end are coupled by capacitors. The input low-pass filter and the low noise preamplifier are coupled by the capacitor C24. The capacitor C23 link the preamplifier output to the band-pass filter and the output amplifier. The C25 separates the output amplifier from the following RF balun. The output

amplifier works in inverting mode and is offset by the resistor divider (R21 and R31). The AC coupling allows to power all the blocks by an asymmetrical power supply.

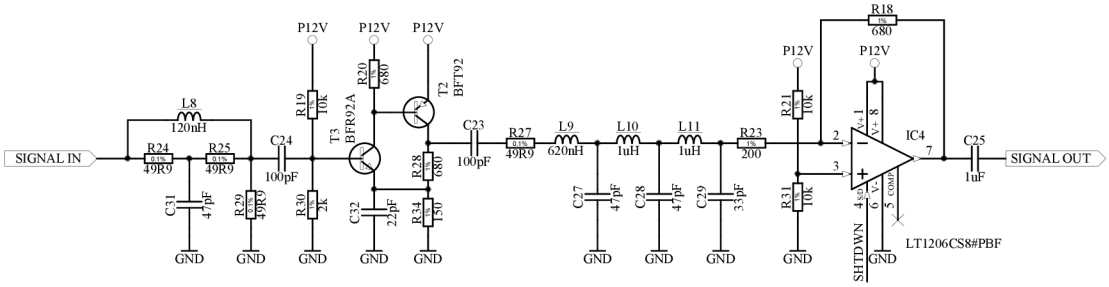



Fig. 2.3: Schematic of the implemented RF Front End

The simulated frequency response of the RF Front End is depicted in Fig. 2.4. Characteristic parameters of the band pass amplifier are the maximum gain of 18.6 dB, the gain at 40 MHz of 16.8 dB, the high-pass cut-off frequency of 19.9 MHz, and the low-pass cut-off frequency of 47.4 MHz.

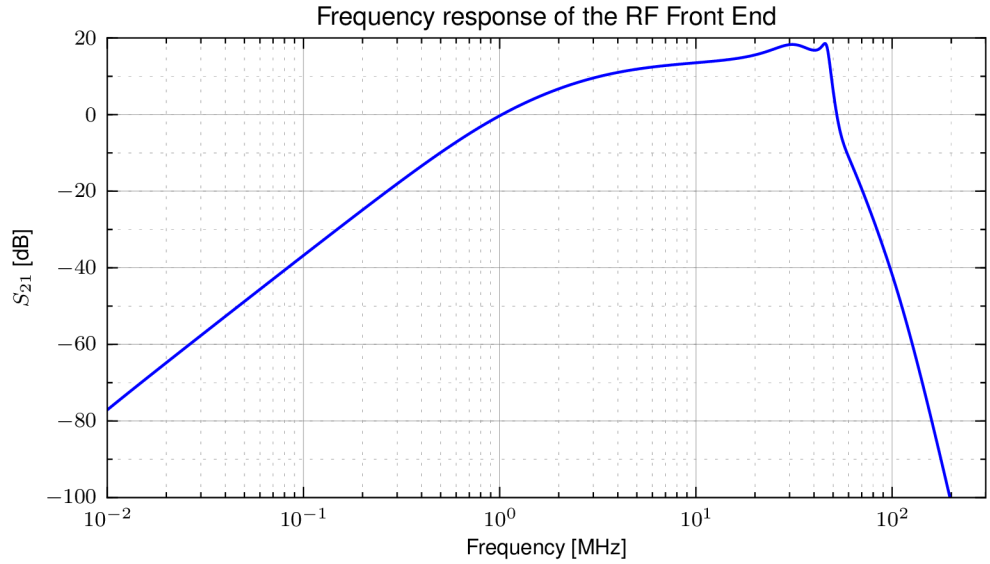



Fig. 2.4: Simulation of the frequency response of the RF Front End

## RF Front End Tolerance Analysis

The simulated frequency response will never be precisely equal to the real circuit characteristics. The real frequency response will deviate simultaneously with a change of component parameters from their nominal values. The possible deviations of circuit characteristics have to be determined by the tolerance analyses. The

observed parameters of the front end tolerance analysis are the gain at frequency of 40 MHz, the maximum gain, and the low cut-off and high cut-off frequencies. The 40 MHz gain defines actively used dynamic range of the ADC. If the used dynamic range was considerably less than maximum range, the digital signal SNR would be degraded. The maximum gain has to be close to the 40 MHz gain. Otherwise it would degrade the system SNR again, because the maximum signal amplitude has to be covered by the full-scale range.

The results of the gain tolerance analysis are shown in Fig. 2.5. The dispersion of the maximum gain is 2 dB and of the 40 MHz gain is 1.5 dB. The variations of the both gains are sufficiently small. In addition the 40 MHz gain variation will be compensated by the system calibration.

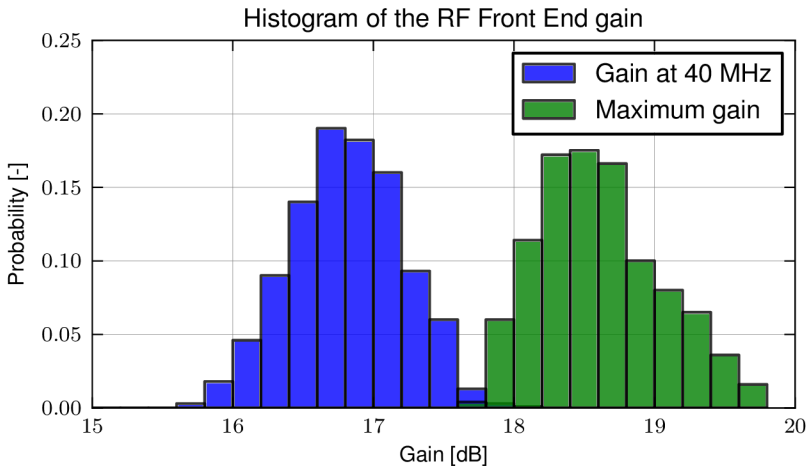



Fig. 2.5: Tolerance analysis of gain of ADC Subsystem RF Front End

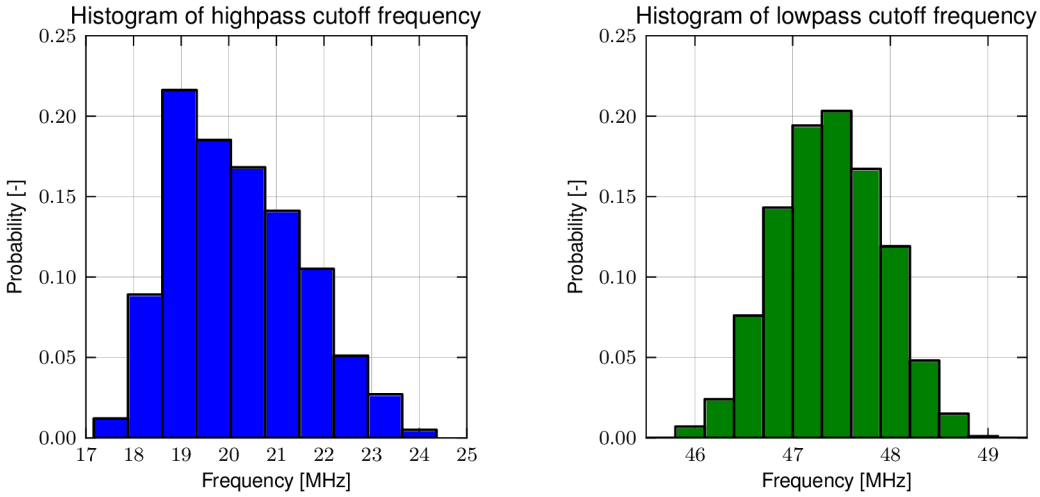
Fig. 2.6 presents the variations of the high-pass low-pass cut-off frequencies of the RF Front End. The high-pass cut-off frequency ranges from 18 MHz to 23 MHz. The range of the low-pass cut-off frequency is from 46 MHz to 48.5 MHz. The variation of the RF Front End pass-band meets the requirements of the undesired signal components filtration.

All the tolerance analyses of the RF Front End give acceptable results. The frequency response of the RF Front End has to be measured for all the finally fabricated ADC Subsystems to verify their functionality and record their exact parameters.

All the tolerance simulations were calculated in PSpice using one thousand runs. Tolerances of all the components were set according to their real parameters. Coils and capacitors have 5 %, resistors with 49.9 Ω have 0.1 %, and the others have 1 %.

### Matching of the ADC Signal Input

The RF Front End output is a single-ended signal, but the following ADC requires a differential signal. In addition the ADC signal input shown in Fig. 2.7 does not




(a) High-Pass cut-off frequency

(b) Low-Pass cut-off frequency

Fig. 2.6: Tolerance analysis of the high-pass and the low-pass cut-off frequencies of the ADC front end

provide the required impedance. Therefore the matching circuit with a balun for single-ended to differential conversion had to be implemented.

The matching circuit significantly influences the quality of the ADC conversion. Therefore it is recommended [15] to adopt the matching circuit from the ADC evaluation module. The matching circuit of the ADS5485 evaluation module [14] is depicted in Fig. 2.8. It is verified by calculation that this particular matching circuit is suitable for the 40 MHz signal matching.

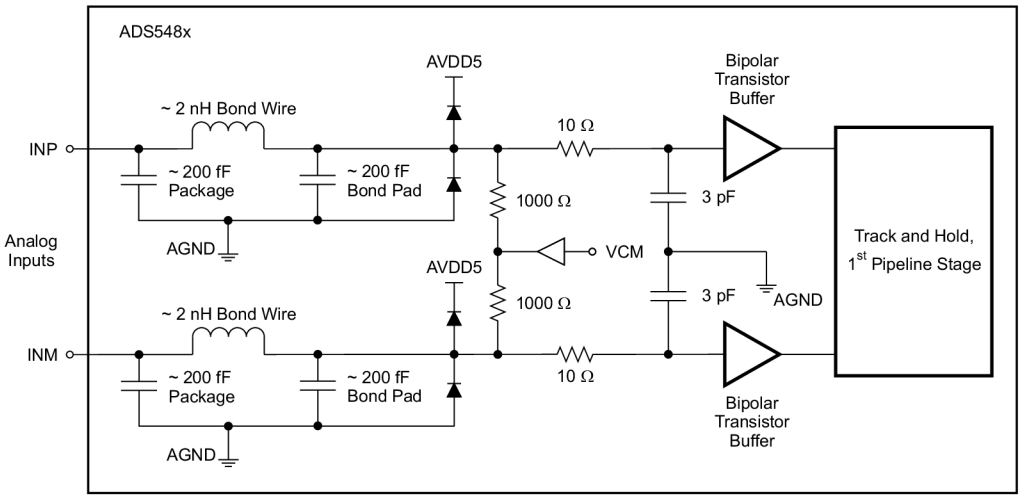



Fig. 2.7: Schematic of the equivalent circuit of the ADS5485 signal input [16]

The used balun ADT1-1WT transforms a signal with the impedance ratio of 1:1. In the ideal situation the required impedance seen by the balun on its secondary

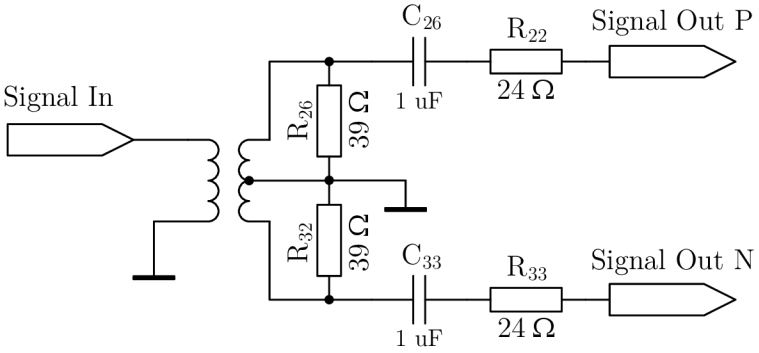



Fig. 2.8: Schematic of the implemented matching circuit for the ADS5485 signal input

side would be 50 Ω to reflect 50 Ω on its primary side. However the transformer is not ideal and it has a non-zero reverse gain. Therefore the compensation has to be calculated by following equations in [15]. The real characteristic impedance of the transformer is calculated using its reverse gain at the signal frequency [17].

$$
\begin{aligned}
s_{12} &= 10^{\frac{-L_{R_{dB}}}{20}} = 10^{\frac{-15.85}{20}} = 0.1613 \\
s_{12} &= \frac{50 - Z_{0_1}}{50 + Z_{0_1}} \\
Z_{0_1} &= 50\,\Omega \cdot \frac{1 - s_{12}}{1 + s_{12}} = 50\,\Omega \cdot \frac{1 - 0.1613}{1 + 0.1613} = 36.11\,\Omega
\end{aligned}
\tag{2.1}
$$

The real impedance on the secondary side $Z_{0_2 sig}$ required to reflect the ideal impedance on the primary side is given

$$
\frac{Z_{0_1}}{Z'_{0_2}} = \frac{Z'_{0_1}}{Z_{0_2}} Z_{0_2} = Z'_{0_2} \cdot \frac{Z'_{0_1}}{Z_{0_1}} = 50\,\Omega \cdot \frac{50}{36.11} = 69.23\,\Omega \tag{2.2}
$$

where $Z'_{0_2}$ is the ideal impedance on the secondary side, and $Z'_{0_1}$ is the ideal impedance on the primary side.

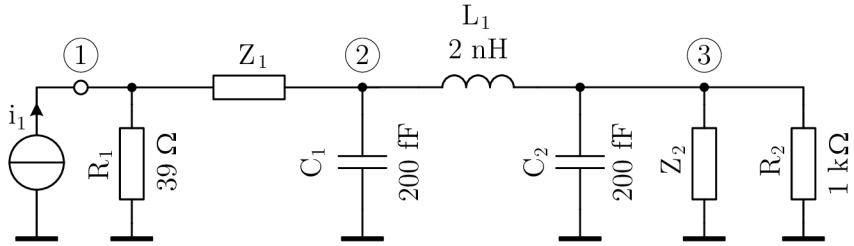



Fig. 2.9: The aggregated circuit used for the input impedance calculation of the matching network and the ADC signal input

The matching circuit (Fig. 2.7) together with the ADC input (Fig. 2.7) has to provide the secondary side with the impedance of $Z_{0_2} = 69.23\,\Omega$. This is verified by the input impedance calculation of the aggregated circuit shown in Fig. 2.9 with respect to ground. Due to the calculus complexity the matrix equation derived from Kirchhoff's laws is used.

$$\begin{pmatrix} G_1 + \boldsymbol{Y_1} & -\boldsymbol{Y_1} & 0 \\ -\boldsymbol{Y_1} & \boldsymbol{Y_1} + \boldsymbol{Y_{C_1}} + \boldsymbol{Y_{L_1}} & -\boldsymbol{Y_{L_1}} \\ 0 & -\boldsymbol{Y_{L_1}} & \boldsymbol{Y_{L_1}} + \boldsymbol{Y_{C_2}} + G_2 + \boldsymbol{Y_2} \end{pmatrix} \begin{pmatrix} \boldsymbol{u_1} \\ \boldsymbol{u_2} \\ \boldsymbol{u_3} \end{pmatrix} = \begin{pmatrix} \boldsymbol{i_1} \\ 0 \\ 0 \end{pmatrix} \tag{2.3}$$

where conductance $G_x$ is the inverse of corresponding resistance $R_x$. The admittance $\boldsymbol{Y_x}$ is the inverse of corresponding impedance $\boldsymbol{Z_x}$. The $\boldsymbol{Y_{C_x}} = j{\cdot}2\pi f C_x$ is admittance of the corresponding capacitor, and $\boldsymbol{Y_{L_1}} = (j{\cdot}2\pi f L_1)^{-1}$ is admittance of the inductor $L_1$.

The input impedance of the circuit (Fig. 2.9) can be calculated from Eq. 2.3 using Cramer's rule and Gaussian elimination.

$$\boldsymbol{Z_{in}} = \frac{\boldsymbol{u_1}}{\boldsymbol{i_1}} = \frac{\boldsymbol{i_1}\frac{(-1)^{1+1}\boldsymbol{\Delta_{1,1}}}{\boldsymbol{\Delta}}}{\boldsymbol{i_1}} = \frac{\boldsymbol{\Delta_{1,1}}}{\boldsymbol{\Delta}} \tag{2.4}$$

where $\boldsymbol{\Delta_{1,1}}$ is a determinant of the reduced admittance matrix from Eq. 2.3 without the first row and the first column, and $\boldsymbol{\Delta}$ is a determinant of the whole admittance matrix. The numerical solution of the input impedance at frequency of 40 MHz is

$$\boldsymbol{Y_1} = \boldsymbol{Z_1}^{-1} = \left(24 + \frac{1}{j \cdot 2\pi \cdot 40 \cdot 10^6 \cdot 10{-}6}\right)^{-1} = (41.67 + 6.908 \cdot 10^{-3} j)\ \text{mS}$$

$$\boldsymbol{Y_2} = \boldsymbol{Z_2}^{-1} = \left(10 + \frac{1}{j \cdot 2\pi \cdot 40 \cdot 10^6 \cdot 3 \cdot 10^{-12}}\right)^{-1} = (5.685 + 753.9j)\ \mu\text{S}$$

$$\boldsymbol{Y_{C_1}} = \boldsymbol{Y_{C_2}} = j \cdot 2\pi \cdot 40 \cdot 10^6 \cdot 200 \cdot 10^{-15} = 50.27j\ \mu\text{S}$$

$$\boldsymbol{Y_{L_1}} = \frac{1}{j \cdot 2\pi \cdot 40 \cdot 10^6 \cdot 2 \cdot 10^{-9}} = -1.989j\ \text{S}$$

$$\boldsymbol{\Delta_{1,1}} = \begin{vmatrix} \boldsymbol{Y_1} + \boldsymbol{Y_{C_1}} + \boldsymbol{Y_{L_1}} & -\boldsymbol{Y_{L_1}} \\ -\boldsymbol{Y_{L_1}} & \boldsymbol{Y_{L_1}} + \boldsymbol{Y_{C_2}} + G_2 + \boldsymbol{Y_2} \end{vmatrix} \tag{2.5}$$

$$\boldsymbol{\Delta} = \begin{vmatrix} G_1 + \boldsymbol{Y_1} & -\boldsymbol{Y_1} & 0 \\ -\boldsymbol{Y_1} & \boldsymbol{Y_1} + \boldsymbol{Y_{C_1}} + \boldsymbol{Y_{L_1}} & -\boldsymbol{Y_{L_1}} \\ 0 & -\boldsymbol{Y_{L_1}} & \boldsymbol{Y_{L_1}} + \boldsymbol{Y_{C_2}} + G_2 + \boldsymbol{Y_2} \end{vmatrix}$$

$$\boldsymbol{Z_{in}} = \frac{\boldsymbol{\Delta_{1,1}}}{\boldsymbol{\Delta}} = \frac{(1.756 - 84.860j) \cdot 10^{-3}}{(0.1159 - 2.259j) \cdot 10^{-3}}$$

$$\boldsymbol{Z_{in}} = (37.50 - 1.15j)\ \Omega$$

$$\boldsymbol{Z_{diff}} = 2\boldsymbol{Z_{in}} = (75.00 - 2.29j)\ \Omega$$

The differential input impedance $\boldsymbol{Z_{diff}} = (75.00 - 2.29j)\ \Omega$ of the matching circuit together with ADC inputs is very close to the calculated required impedance

$Z_{0_2} = 69.23\,\Omega$ (Eq. 2.2). The difference can be caused by using the equivalent ADC input circuit, which does not have to equal the real ADC input in the whole range of frequencies, or by the balun leakage inductance, which has not been taken into consideration.

The verified matching circuit was implemented and measured in the ADC Subsystem. Its measurement shows the input reflection of -34.8 dB at 40 MHz which is sufficient for this application.

### 2.1.3 ADC Clock Generation

The signal provided by the RF Front End has to be sampled synchronously with the beam. This is ensured by the bunch clock reception and sampling clock generation. The generation of the sampling clock is depicted in Fig. 2.10.

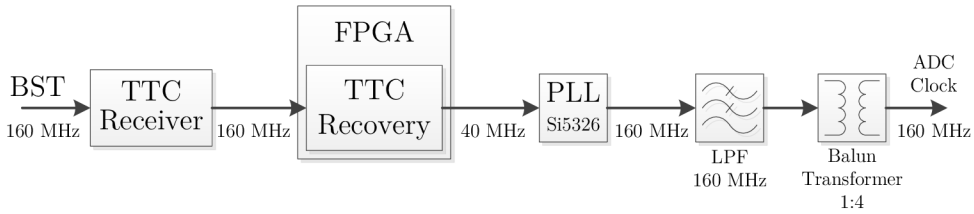



Fig. 2.10: Block diagram of the sampling clock generation

The FPGA recovers the 40 MHz bunch clock from the TTC clock signal. The bunch clock is routed to drive the PLL. The PLL multiplies the bunch clock frequency four times to obtain sampling clock frequency of 160 MHz. The generated sampling clock signal is phase-locked to the LHC bunch clock, thus the ADC sampling and the beam are coherent. The phase-locked sampling clock could be generated by the PLL in the FPGA, but this solution does not achieve a required clock jitter of less than 1 ps. Therefore the PLL Si5326 is used to attenuate jitter of the sampling clock down to 300 fs.

### 2.1.4 Clock Low-Pass Filter

The sampling clock generated by the PLL Si5326 is in the form of a square-wave. The ADC requires the sine-wave clock signal to achieve the best signal to noise ratio [16]. Therefore the sampling clock is filtered by a low-pass filter to remove all its higher harmonic components.

The clock low-pass filter design is not part of this thesis. It is examined to provide a complete view on the ADC Subsystem. The structure and component values of the filter have been adopted from the first FBCCM prototype. The filter structure is depicted in Fig. 2.11. The filter is a $12^{\text{th}}$ order elliptic filter consisting of two equal filter stages in series.

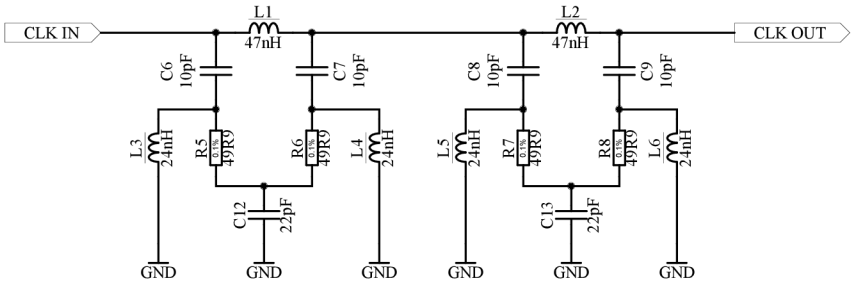



Fig. 2.11: Schematic diagram of ADC clock filter

The simulated frequency response of the clock filter is shown in Fig. 2.12. The filter was loaded with its characteristic load of 50 Ω. It can be seen that the filter cut off frequency is 190 MHz and the first significant minimum of the insertion gain is at the frequency of 325 MHz. The filter frequency characteristic could be more suitable to filter the 160 MHz square wave clock signal, because the main unwanted component is the clock third harmonic at the frequency of 480 MHz. If it is discovered in the future tests that the purity of the clock signal is not sufficient, the new filter with a zero of the transmission at a frequency around the third harmonic will be designed.

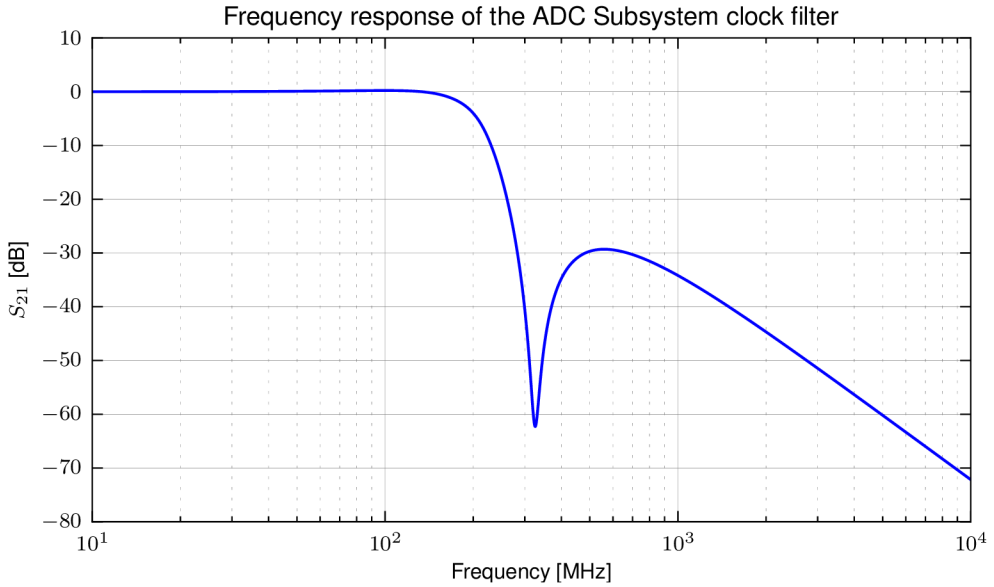



Fig. 2.12: Simulated frequency response of the Clock Low-Pass Filter

### Clock Low-Pass Filter Tolerance Analysis

The frequency responses of simulated filter and the real filter will never be same. Deviation of the real filter frequency response is determined by the tolerance analysis. The filter tolerance analysis was performed in PSpice. Results of this analysis are the tolerance of the low cut-off frequency (Fig. 2.13) and the tolerance of filter gain at the clock frequency of 160 MHz (Fig. 2.14).

The tolerance analysis was performed with one thousand repetitions and the uniform distribution of the component value probability. The tolerances of the components from the schematic in Fig. 2.11 were set according to characteristics defined by their manufacturers. All the inductors have 2 % tolerances, all the capacitors have 5 %, and the resistors have 0.1 %.

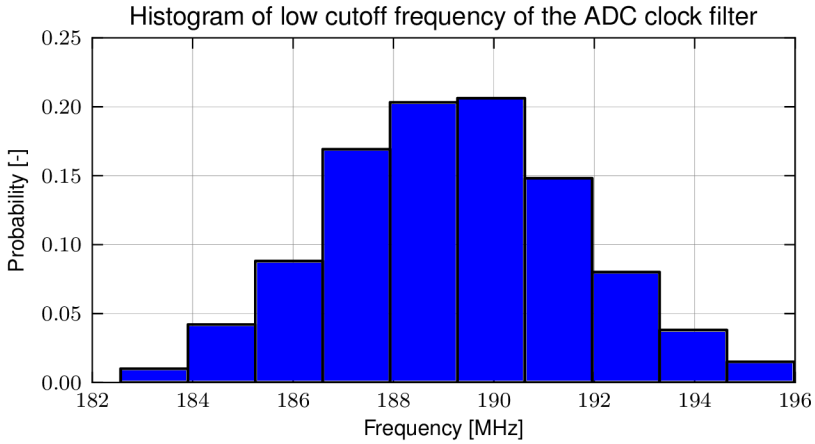



Fig. 2.13: Simulation of the low cut-off frequency tolerance of the Clock Low-Pass Filter

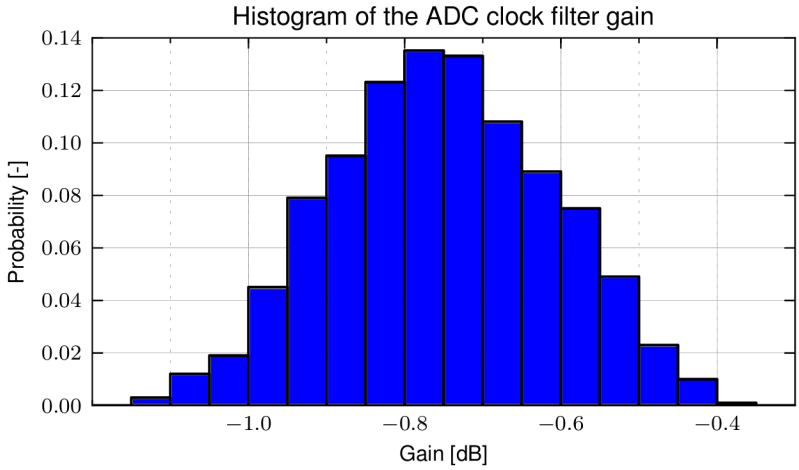



Fig. 2.14: Simulation of the gain tolerance of the Clock Low-Pass Filter

The low cut-off frequency varies from 182 MHz to 196 MHz. This is sufficient interval, because the input clock signal has only the wanted spectral line at frequency of 160 MHz. The low cut-off frequency of the filter is not critical parameter unless it is close to the clock frequency. The filter gain at the frequency of 160 MHz is more important parameter, because the amplitude of the clock signal injected into the ADC should be in range from 6 dBV to 14 dBV. The tolerance analysis shows the most gains in the interval from -1 dB to -0.5 dB. The spread of the filter gain is sufficient for the final fabrication, because the range of the tolerated clock amplitude is wider.

### 2.1.5 Single-Ended to Differential Clock Conversion

The filtered clock generated by the Si5326 is a single-ended signal with maximum amplitude of 2.5 $V_{PP}$ [18]. The ADC requires a differential clock with amplitude in the range from 3 $V_{PP}$ to 5 $V_{PP}$ [16]. Therefore the clock for the ADC provided by a clock low-pass filter is gained and converted to differential signal by a balun transformer.

The required gain of the balun transformer had to be calculated. The recommended input voltage is in the range from $V_{Imin_{dB}} = 6$ dBV to $V_{Imax_{dB}} = 14$ dBV. The PLL clock output level is in the range from $V_{Omin_{dB}} = 4$ dBV and $V_{Omax_{dB}} = 8$ dBV. The insertion loss of the clock low-pass filter is $L_{LPF} = 0.8$ dB. The desired voltage gain of the balun transformer is solved using an equation

$$G_B - L_B = V_{O_{dB}} - V_{I_{dB}} + L_{LPF} \tag{2.6}$$

where $G_B$ is the gain of the balun, $L_B$ is the insertion loss of the balun, $V_{O_{dB}}$ is the PLL output voltage, and $V_{I_{dB}}$ is the ADC input voltage in decibels. The minimum voltage gain of the balun transformer is given

$$G_{B_{min}} - L_B = V_{Imin_{dB}} - V_{Omin_{dB}} + L_F = 2.8 \text{ dB}. \tag{2.7}$$

The maximum voltage gain is given

$$G_{B_{max}} - L_B = V_{Imax_{dB}} - V_{Omax_{dB}} + L_F = 6.8 \text{ dB}. \tag{2.8}$$

Due to the calculated range the balun transformer with $G_B = 6$ dB can be used. The voltage ratio of such transformer is 1:2, thus the transformer impedance ratio is 1:4. The chosen transformer gain corresponds to gain of the transformer on the ADS5485 evaluation module [14]. The used balun transformer is ADT4-1WT with insertion loss $L_B = 1$ dB at 160 MHz [19].

### 2.1.6 Clock Termination

The characteristic impedance at the balun transformer input has to be 50 Ω, because the clock low-pass filter requires to be loaded with this nominal impedance. This is ensured by the matching network on the transformer secondary side. The matching

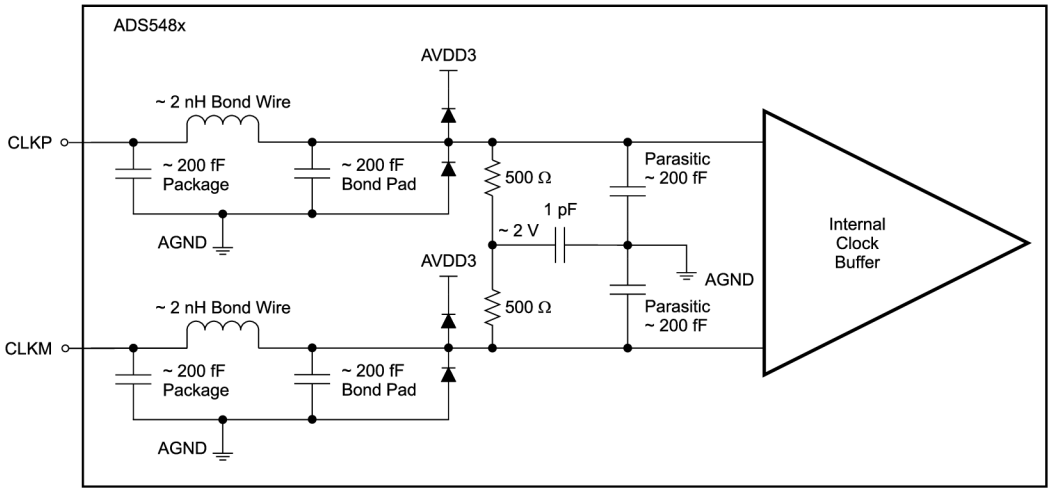



Fig. 2.15: Equivalent ADS5485 clock input circuit [16]

network has to be designed with respect to the impedance of the ADC clock input. The equivalent circuit of the ADC clock input is shown in Fig. 2.15.

The required impedance seen by the ideal RF balun transformer would be 200 Ω to reflect 50 Ω on the transformer input side. The calculation of the matching circuit for the real transformer has to reflect its non-zero reverse gain [19]. The procedure of the impedance matching calculation is similar like for the ADC signal input [15]. The real impedance $Z_{0_1}$ on the primary side of the transformer is calculated.

$$\begin{aligned} s_{12} &= 10^{\frac{-L_{R_{dB}}}{20}} = 10^{\frac{-20}{20}} = 0.1 \\ s_{12} &= \frac{50 - Z_{0_1}}{50 + Z_{0_1}} \\ Z_{0_1} &= 50\,\Omega \cdot \frac{1 - s_{12}}{1 + s_{12}} = 50\,\Omega \cdot \frac{1 - 0.1}{1 + 0.1} = 40.91\,\Omega \end{aligned} \tag{2.9}$$

The impedance $Z_{0_2}$ which the transformer has to see on the secondary side to reflect 50 Ω into the primary side is given by the equation.

$$\begin{aligned} \frac{Z_{0_1}}{Z'_{0_2}} &= \frac{Z'_{0_1}}{Z_{0_2}} \\ Z_{0_2} &= Z'_{0_2} \cdot \frac{Z'_{0_1}}{Z_{0_1}} = 200\,\Omega \cdot \frac{50}{40.91} = 244.4\,\Omega \end{aligned} \tag{2.10}$$

The input differential impedance of the matching circuit and the ADC clock input has to be 244.4 Ω. The calculation of the matching circuit input impedance requires knowledge of the ADC clock input impedance. There are three basic ways how to solve the ADC clock input impedance (Fig. 2.15). The first way is analytic which means to calculate the impedance using any circuit solving method. Another way is to simulate the input circuit in circuit analyser program (e.g. PSpice). The

last way is to use the ADC scattering parameters declared by the manufacturer which should be the most precise definition of the real impedance. The scattering parameters are not available for the used ADC. Therefore the input impedance is calculated.

The input impedance calculation is limited to the high frequency signals. The equivalent clock input circuit (Fig. 2.15) is simplified to schematic diagram depicted in Fig. 2.16 for the calculation of input impedance with respect to ground (AGND). The complex impedance of the clock input with respect to ground can be calculated using the following equations.

$$\begin{aligned} \boldsymbol{Y_1} &= R_1^{-1} + j \cdot 2\pi f(C_2 + C_3) = (2 + 0.402j)\ \text{mS} \\ \boldsymbol{Y_{L_1}} &= (j \cdot 2\pi f L_1)^{-1} = -0.497j\ \text{S} \\ \boldsymbol{Y_{clk}} &= \frac{\boldsymbol{Y_1 Y_L}}{\boldsymbol{Y_1 + Y_L}} + j \cdot 2\pi f C_1 = (2.003 + 0.5954j)\ \text{mS} \\ \boldsymbol{Z_{clk}} &= \boldsymbol{Y_{clk}}^{-1} = (458.67 - 136.34j)\ \Omega \end{aligned} \tag{2.11}$$

where the $\boldsymbol{Y_1}$ is complex admittance of the parallel combination of C2, R1, and C3. The $\boldsymbol{Y_{L_1}}$ is complex admittance of L1. The $\boldsymbol{Y_{clk}}$ and the $\boldsymbol{Z_{clk}}$ are complex admittance and impedance of the clock input with respect to ground.

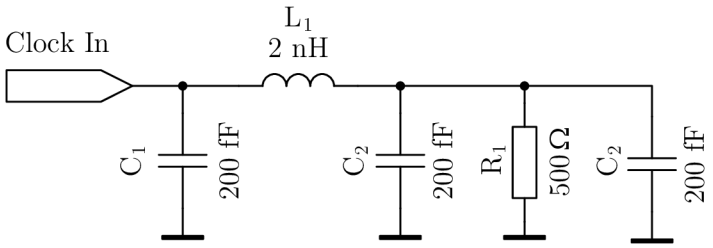



Fig. 2.16: Schematic of the equivalent simplified circuit of the ADS5485 clock input pin

The easiest way to provide an external matching is to connect external combination of resistive and reactive components parallel to the clock input. The parallel admittance of the clock input and the matching circuit have to comply with the following equation to produce the required impedance on the secondary side of the balun.

$$\begin{aligned} Y_{0_2} &= \frac{\boldsymbol{Y_{clk}}}{2} + \boldsymbol{Y_{cm}} \\ \boldsymbol{Y_{cm}} &= Z_{0_2}^{-1} - \frac{\boldsymbol{Y_{clk}}}{2} \\ \boldsymbol{Y_{cm}} &= 244.4^{-1} - 0.5 \cdot (2.003 + 0.5954j) \cdot 10^{-3} = (3.090 - 0.2977j)\text{mS} \\ \boldsymbol{Z_{cm}} &= \boldsymbol{Y_{cm}}^{-1} = (320.6 + 30.89j)\ \Omega \end{aligned} \tag{2.12}$$

where $Y_{0_2}$ is the required characteristic admittance on the secondary side derived from eq. 2.10 and $\boldsymbol{Z_{cm}}$ is impedance of the designed matching circuit for the clock lines. The clock input admittance in Eq. 2.11 is calculated with respect to ground, but in Eq. 2.12 admittances are differential. The clock input admittance is therefore divided by two to obtain the differential admittance of the clock inputs.

The parallel matching load has to be a serial combination of a resistor and an inductor, because the reactive part of the matching impedance is positive. The result is compliant with an expectation that an inductor has to compensate the capacitive character of the ADC clock input. The matching load could be a parallel combination of a resistor and an inductor as well, but the inductance of the parallel combination would be excessively high for the clock frequency. The resistance of the matching resistor $R_{cm}$ equals the real part of the calculated impedance (Eq. 2.12) and the inductance of the serial inductor $L_{cm}$ is derived from its imaginary part.

$$\begin{aligned} R_{cm} &= \mathrm{Re}(\boldsymbol{Z_{cm}}) = 442.8\,\Omega \\ L_{cm} &= \frac{\mathrm{Im}(\boldsymbol{Z_{cm}})}{2\pi f} = \frac{30.89}{2\pi \cdot 160 \cdot 10^6} = 30.7\ \mathrm{nH} \end{aligned} \tag{2.13}$$

The designed matching circuit with the RF balun transformer is depicted in Fig. 2.17. It was tested on the ADC evaluation module. The input reflection was measured to be -16 dB which is sufficient matching for the clock low-pass filter.

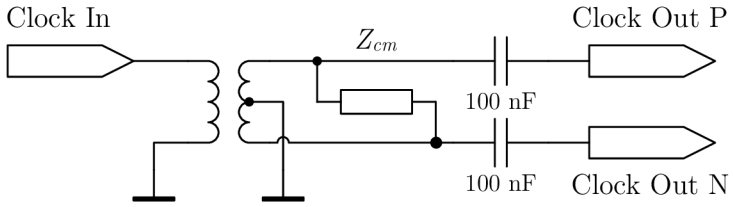



Fig. 2.17: Balun transformer and matching load of the ADC clock signal

### 2.1.7 TTC Receiver

The bunch clock, which the PLL is provided with, is recovered in the FPGA from the TTC signal. The TTC signal is distributed as an optical signal through the optical fibre. The optical signal from the fibre is converted by an optical SFP module. The optical SFP module is a module commonly used in telecommunications. It integrates a reciever and transmitter. Only the receiver is used in the FBCCM as no uplink is required.

All the suitable SFP modules comply with the same standards, thus the TTC Receiver design can be independent of the particular SFP module. The currently used SFP module in the FBCCM is the AFCT-5701PZ. Advantageously it could be exchanged to another optical module in the future if the optical receiver parameters (e.g. wavelength or sensitivity) were redefined.

The electrical signal provided by the SFP module is an asynchronous serial data stream. It cannot be processed directly by the FPGA Cyclone III, because this particular FPGA is not equipped with any clock recovery circuit. A dedicated circuit ADN2814 is used for clock recovery from the serial data stream. This circuit contains a PLL which locks on to the input serial stream. The output of the PLL is used as clock signal for the FPGA. This method with PLL is necessary for clock recovery, because the TTC clock is used for the beam synchronous sampling. The recovered clock and the data are used for recovery of the 40 MHz bunch clock in the FPGA. The recovery is based on the decoding of the bi-phase mark code.

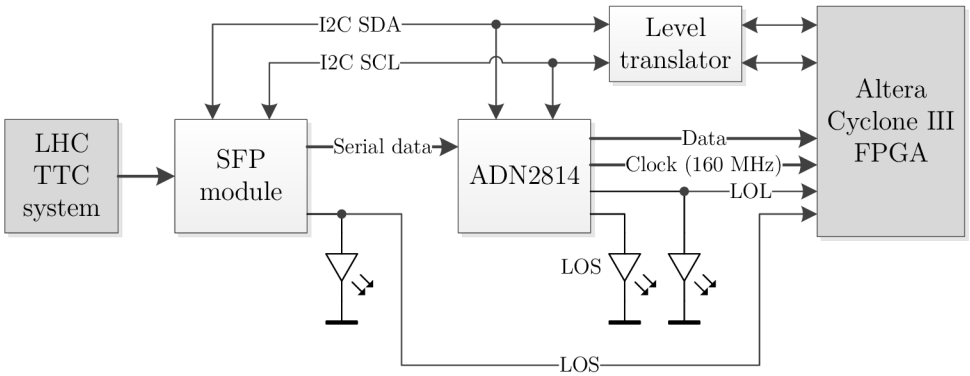



Fig. 2.18: Block diagram of the TTC reciever

The block diagram of the TTC receiver is depicted in Fig. 2.18. The TTC clock signal is connected to dedicated FPGA pins thus the signal can be connected to clock nets inside the FPGA effectively. This is required for high performance of the FPGA firmware. The HSMC connector pins are connected to the 2.5 V FPGA banks. The TTC receiver uses the 3.3 V CMOS standard. The TTC data and clock signals do not require any voltage level translation, because both signals are differential LVDS. The voltage level translation is implemented on the loss of lock (LOL) and the loss of signal (LOS), although it is not drawn in the picture. The SFP module and the clock recovery circuit share the management $I^2C$ bus. The signals of this bus require bidirectional voltage level translators.

The bidirectional voltage level translation between 3.3 V and 2.5 V logic for $I^2C$ is performed by the circuit depicted in Fig. 2.19. The circuit contains a N-channel MOSFET and two pull-up resistors. A bipolar transistor cannot be used for this circuit. The internal MOSFET diode pulls down the FPGA side of the bus to ground when any circuit of the TTC side pulls down the bus. The chosen MOSFET has to be sensitive enough for gate-to-source voltage less than 2.5 V, otherwise the circuit would not work. The MOSFET transistor BSS138 is widely used for these purposes.

### 2.1.8 Indicators and Controls

On the front panel (Fig. 2.20) all important indicators and controls have to be visible and accessible to clearly indicate the device status.

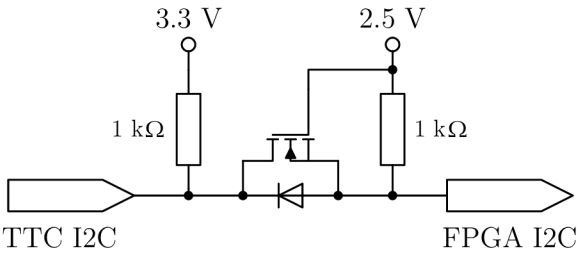



Fig. 2.19: Bidirectional voltage level translator for $I^2C$

There are three LEDs showing the TTC receiver status (already depicted in Fig. 2.18). Two of them indicate the loss of signal. The loss of signal provided by the SFP module reflects the optical signal level. The loss of signal provided by the ADN2814 corresponds to the electrical signal level of the SFP module. The third LED is active when the PLL in the ADN2814 is not locked on to the input signal. These three LEDs are driven by the TTC receiver hardware, thus their functions cannot be modified.

The other LEDs are driven by the FPGA and in general indicate the status of the FBCCM device. Two LEDs show settings of beam permit flags for the CIBU. Other LEDs blink when the BeagleBone registers an IRQ or when a message is sent to the Ethernet network. The last one indicates an error of telegram reception.

Besides the indicators, two push buttons are placed on the front panel. Both of them are connected to the FPGA. The first button resets the the FPGA and the BeagleBone and the other starts the test of the Interface for CIBU.

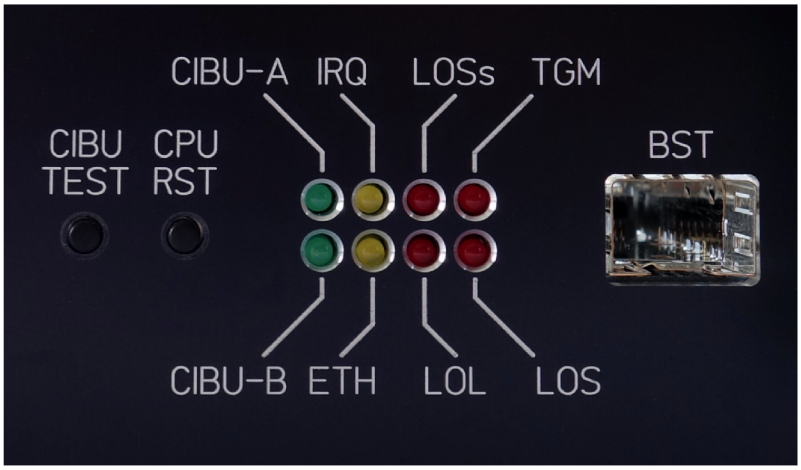



Fig. 2.20: Picture of the front panel indicators and controls

### 2.1.9 Serial Number

The silicon serial number chip DS2411 provides the subsystem with an unique serial number. It is connected by a one-wire data bus to the FPGA. The serial number allows to unambiguously identify each of fabricated boards, e.g. in measurement reports.

## 2.2 CPU Subsystem

The CPU Subsystem provides different functionality than the ADC Subsystem. It implements exclusively digital interfaces. The block diagram of the CPU Subsystem is depicted in Fig. 2.21. Its complete schematic diagram is shown in Appendix A.1.

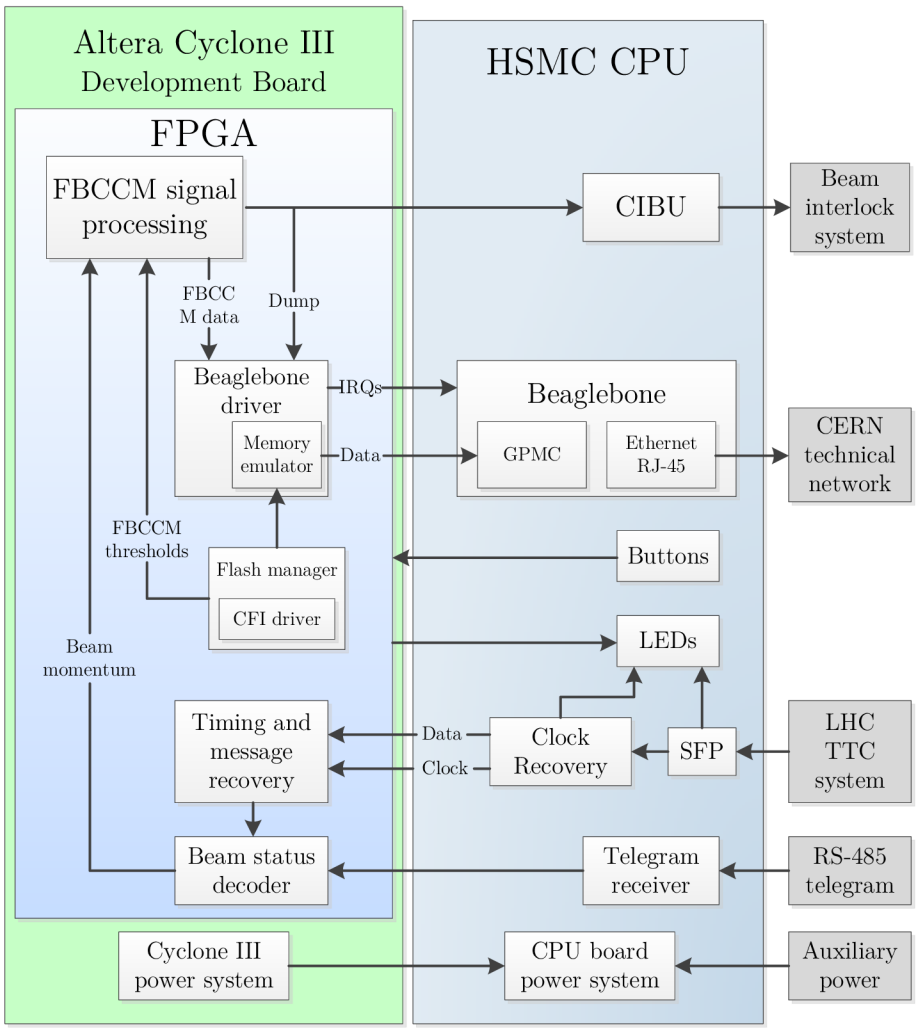



Fig. 2.21: Block diagram of the CPU Subsystem

The CPU Subsystem contains the TTC receiver like the ADC Subsystem in order to have an option to connect the optical fibre from the LHC timing system

either to the front or back panel depending on a particular FBCCM installation. The equivalent indicators and controls are placed on both front and back panels (respectively in the ADC Subsystem and the CPU Subsystem) to be easily accessible from both sides of the installed FBCCM device.

The TTC receiver, indicators and controls are not examined in this section, because they have been already described in the section 2.1.7 TTC Receiver.

### 2.2.1 BeagleBone

The CPU Subsystem provides the FBCCM with the Ethernet interface to CERN technical network. This interface is represented by the BeagleBone.

The BeagleBone is a microprocessor module based on the Sitara AM335x ARM Cortex-A8 Microprocessor. The microprocessor can run at frequency up to 720 MHz, and it has 256 MB of DDR2 memory available. It is a very powerful module equipped with the Ethernet interface. It can provide a connection for few clients, and potentially run a web server with the data history.

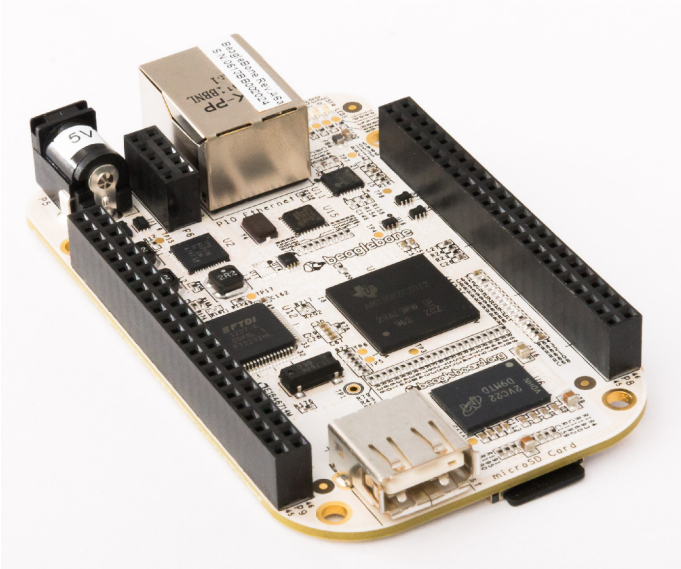



Fig. 2.22: BeagleBone microprocessor module

Concerning the other available peripherals on the BeagleBone, there is one USB port and a slot for the SD card. In addition there are two expansion connectors of 46 pins each. Their pins are routed to the processor and serve to many purposes depending on the processor configuration. These connectors make the interfacing the FPGA simple. They contain an interface for the processor initialisation which allows the FPGA to load the software into the processor.

The BeagleBone is an industrial module and the Sitara AM335x ARM Cortex-A8 Microprocessor has a long term support. Regarding its characteristics and the

facts that all data sheets are available, the BeagleBone is highly suitable for this application.

The detailed description of the BeagleBone including device schematics can be found in [20]. The connection of the BeagleBone is very undemanding. If the SD card was used for loading of the main application, the BeagleBone would need to connect just 5 V power through either the expansion connectors, the power connector or the USB. The program of the BeagleBone for the FBCCM has to be stored in the flash memory of the Cyclone III Development Board in order to be easily upgraded remotely by the Altera Ethernet Blaster. Therefore pins that affect settings of the boot mode, called boot pins, and pins of an interface for the bootloader loading have to be connected to the FPGA too. Even though it would be sufficient to connect only these necessary pins of the BeagleBone to the FPGA, all the pins of expansion connectors are connected to the FPGA. These connections provide freedom in the future development.

### 2.2.2 Client Interface for CIBU

The CPU Subsystem implements a client interface for the CIBU. The CIBU has been described in the section 1.5.1 User Interface of the BIS.

The circuit depicted in Fig. 2.23 drives the current exceeding 9 mA through the CIBU to permit the beam. In the CPU Subsystem there are two exactly same circuits. Each of them is independently driven by the FPGA to comply with the CIBU standard [11].

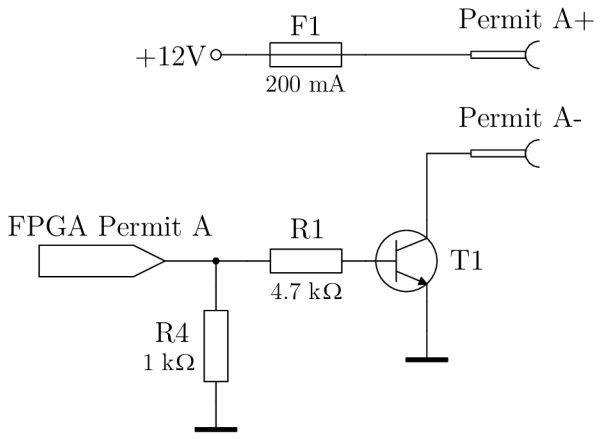



Fig. 2.23: Implemented output circuit for the client CIBU interface

The resistance of the R4 in the output circuit has to be low enough to dump the beam by default. Otherwise the pull-up resistor activated during the FPGA loading could provide enough current for the transistor T1 to permit the beam inadvertently.

The back panel connector for the CIBU is a circular connector Burndy with eight pins. This connector cannot be soldered directly to the PCB of the CPU Subsystem.

Therefore the connector for the CIBU is mounted on the back panel and connected to the CPU Subsystem with flexible flat cable as it is shown in Fig. 2.24.

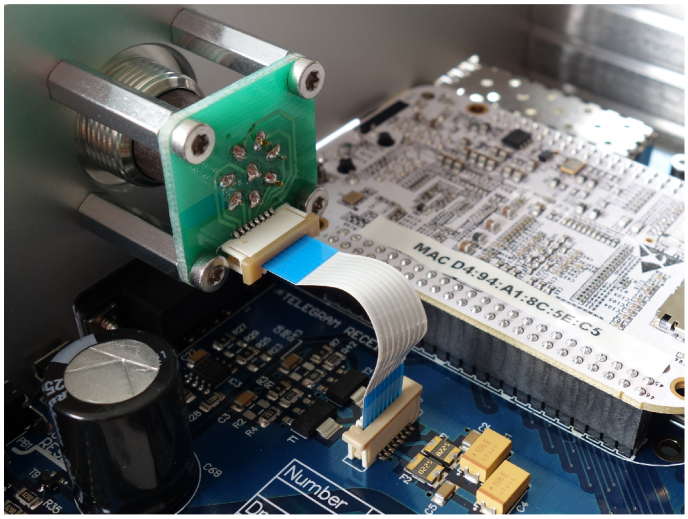


Fig. 2.24: Flexible connection between the CPU board and the back panel connector for the CIBU

### 2.2.3 Clock Input and Output, Telegram Receiver

The CPU Subsystem implements the clock input and output, and the telegram receiver as a potential replacement of the TTC optical fibre when the optical link is not available. Therefore the clock input and output are designed to operate with signals of frequencies at least 40 MHz (bunch clock frequency).

#### Clock Input

The clock input circuit (Fig. 2.25) is based on a buffer with hysteresis. The chosen buffer is SN74LVC1G17, because it accepts power voltage of 2.5 V. This ensures output levels 0 V for a logic low level and 2.5 V for a logic high level. Otherwise a voltage level translator would have to be used to provide the FPGA input with required voltage levels. The buffer is protected by a transient-voltage-suppression (TVS) diode and an input resistor of 100 Ω in series. The diode has to be specially designated for high speed interfaces (e.g. USB), otherwise the capacitance of the diode would pull down the input signal. The well suitable diode for this design is SRV05-4 with the capacitance less than 3 pF. The 49.9 Ω resistor provides the input impedance matching.

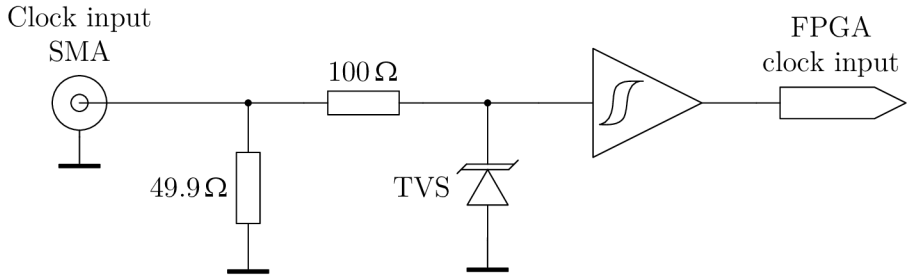



Fig. 2.25: Circuit for the clock input of the CPU board

### Clock Output

The clock output is required to be back-matched and withstand a short circuit outside the device without damaging the FPGA. Although normally it would be sufficient to route the output FPGA signal directly to the output, due to the mentioned requirements a back-matching serial resistor of the characteristic impedance is inserted into the output line.

The back-matching resistor causes attenuation of 6 dB which is compensated by a non-inverting amplifier. The back matching and the amplifier are depicted in Fig. 2.26.

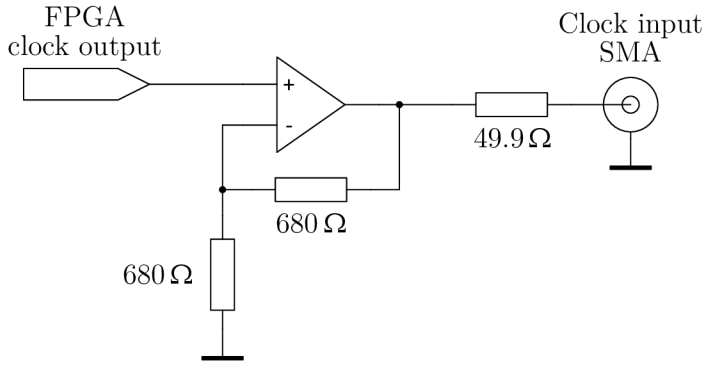



Fig. 2.26: Schematic of the clock output amplifier of the CPU Subsystem

The current-feedback operational amplifier THS3091 provides sufficient bandwidth of 135 MHz with the required gain of 6 dB. The DC-coupling of the clock output is required and no available rail-to-rail amplifier provides the sufficient bandwidth. Consequently the operational amplifier has to be supplied by a symmetric power supply.

### Telegram Receiver

The CPU System without TTC signal would have to acquire the beam energy and the UTC time from the GMT telegram (Sec. 1.4 General Machine Timing). The

receiving circuit for the RS-485 of the GMT is SN65HVD3088. The output of the receiver is connected to the FPGA through a resistor divider, because the output level of the chip is 5 V and FPGA requires the 2.5 V signal levels. The back panel connector for this interface is the commonly used Canon DB-9 connector.

## 2.3 FBCCM Electric Power Distribution System

The electric power distribution system of the FBCCM is depicted in Fig. 2.27. The FBCCM is supplied by the mains distribution network. The mains electricity is connected through the entry power module FN1393-10-05-11 from Schaffner. This power line filter in the entry module diminishes interference entering the FBCCM box over power lines. At the same time the entry module provides the FBCCM with the basic short-circuit protection and the main device switch. The filtered mains electricity supplies two AC/DC power converting modules TXL 060-15S from Tracopower.

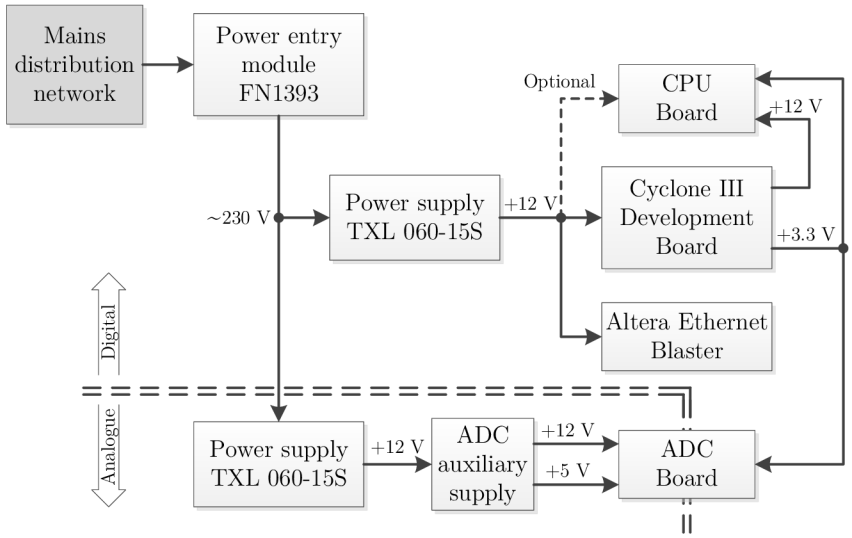



Fig. 2.27: FBCCM power system

The first power module supplies exclusively the digital part of the design which includes the FPGA board, the Altera Ethernet Blaster, the CPU Subsystem, and partially the ADC Subsystem. The other Tracopower module powers entirely the analogue part of the ADC Subsystem. The nominal output voltage of the Tracopower module is 15 V. All the electronic subsystems are designed to operate with power voltage of 15 V, although the nominal power voltage for these subsystems is 12 V.

All the power supplies and the entry power module are geometrically and electromagnetically separated from the other electronics in the FBCCM box. The electromagnetic separation is provided by a thick aluminium bar. The separation should

break off any potential electromagnetic interference that could come from the switching power supplies or mains distribution wires. Otherwise the interference could negatively influence the performance of the RF Front End and the ADC.

### 2.3.1 ADC Auxiliary Power Supply

The ADC Auxiliary Power Supply provides the ADC Subsystem with an extra filtered voltage of 12 V and regulated voltage of 5 V. The voltage of 5 V is derived from the voltage of 12 V provided by Tracopower module.

The block diagram of this power supply is depicted in Fig. 2.28. The desired maximum load of the 5 V output is 3 A, the maximum current of the input is 4 A. The current of the 12 V output is around 2.4 A depending on the load of the 5 V output. The power input and outputs of the supply are protected by the resettable PTC fuses. Their ratings correspond to stated maximum currents.

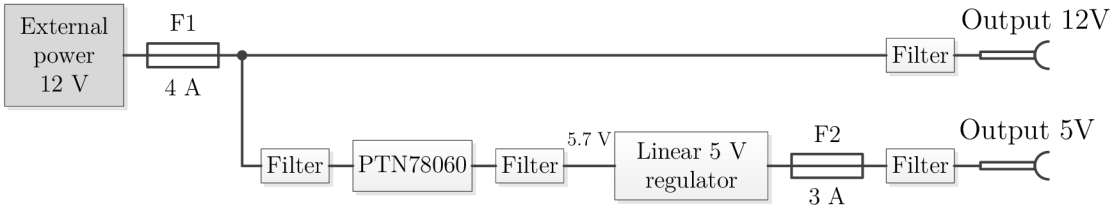



Fig. 2.28: Block diagram of the ADC Auxiliary Power Supply

The voltage regulation of the 12 V input to 5 V output is performed in two stages. The first regulator a switching mode integrated regulator PTN78060. It only requires an external resistor for the output voltage setting. The regulator's input and output are filtered to suppress interference coming from the switching. The filtration of the regulated voltage is improved by the following linear regulator. The linear regulator is a LDO regulator LT1764A with very low output voltage noise. The voltage drop out of the linear regulator is less than 600 mV [21]. Therefore the switching regulator output voltage is set up to 5.7 V.

The two stage regulation has been designed to minimise losses of the linear regulator. The lost power in the design with an one stage linear regulation would be

$$P_{L_{LR}} = (U_{1_{max}} - U_2) \cdot I = (15 - 5) \cdot 3 = 30 \text{ W}. \tag{2.14}$$

The losses of the linear regulator in the two stage design are

$$P_{L_{2st}} = (U_1' - U_2) \cdot I_{max} = (5.7 - 5) \cdot 3 = 2.1 \text{ W} \tag{2.15}$$

which is considerably less than 30 W losses of the stand-alone linear regulator. The total lost power of this solution is

$$P_{L_{tot}} = (1 - \eta) \cdot U_1' \cdot I_{max} + PL_{2st} = (1 - 0.9) \cdot 5.7 \cdot 3 + 2.1 = 3.8 \text{ W} \tag{2.16}$$

including the switching mode regulator efficiency of 90 % for the load current of 3 A [22]. These losses are acceptable in comparison with the losses of the pure linear regulation.

The noise of power supplies is taken very seriously. The 5 V output is again filtered to suppress high frequency noise. The switching regulator is placed into the copper shielding box to limit electromagnetic emission.

The complete schematic diagram of the ADC auxiliary power supply can be found in Appendix A.1. The photographs of the implemented ADC auxiliary power supply in Appendix A.2.3.

### 2.3.2 Power Distribution of the ADC Subsystem

The power distribution of the ADC Subsystem is depicted in Fig. 2.29. The supplies powering digital and analogue electronics are separated, because the digital electronics generally emit higher level of noise interference due to switching transient effects. The separation limits the negative influence of the noisy digital electronics on the sensitive analogue circuits.

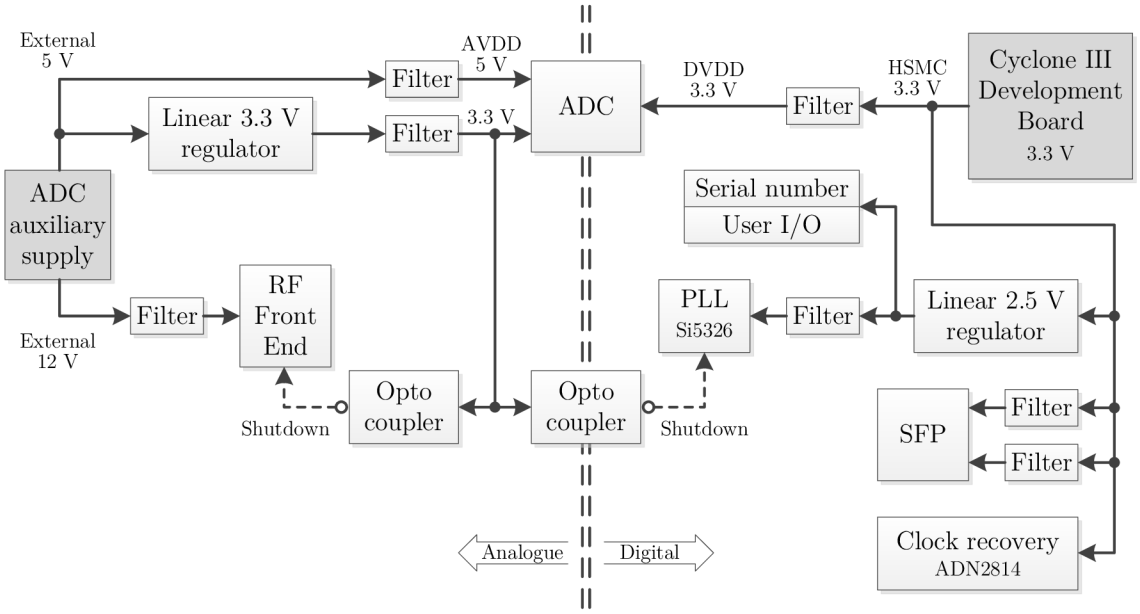



Fig. 2.29: Block diagram of power distribution of the ADC Subsystem

The used ADC ADS5485 requires three different power voltages [16]. The power voltages of 3.3 V and 5 V are required on the analogue side. The 5 V power voltage is provided by the ADC Auxiliary Power Supply. The 3.3 V power voltage is derived from the 5 V power voltage by the linear 3.3 V regulator. The digital side of the ADC is powered by the 3.3 V power voltage provided by the Cyclone III Development Board. All the power lines entering the ADC are filtered to minimise the power voltage noise.

ADC Auxiliary Power Supply provides the RF Front End with 12 V power voltage. This power voltage in the ADC Subsystem is independent of the ADC power voltages. The ADC could be damaged if the ADC without analogue side power voltages was provided with the input signal or the clock signal. Therefore RF Front End and PLL Si5326 have to be shut down when the ADC is not powered. These circuits are equipped with a dedicated shutdown inputs. They are driven by the optocoupler circuit (Fig. 2.30) to comply with the separation of the power distribution.

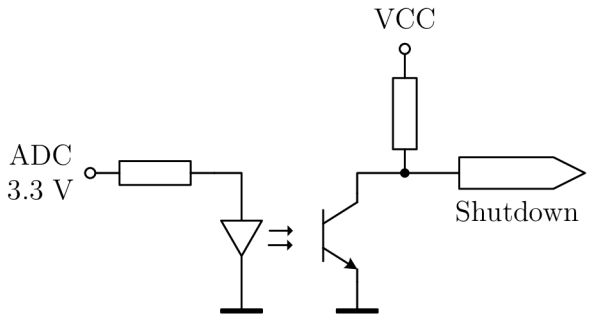



Fig. 2.30: Schematic of the optocoupler shutdown circuit

The PLL Si5326 is powered by 2.5 V power voltage provided by the linear 2.5 V regulator. It could be powered directly by the Cyclone III Development Board voltage of 3.3 V, but the standard of the PLL pins would not comply with the connected FPGA bank. This linear 2.5 V regulator in addition supplies the serial number due to the same reason. The Cyclone III Development Board powers this 2.5 V regulator and circuits of the TTC Receiver as well.

All the filters in the power distribution are $2^{nd}$ order low-pass $\pi$-filters. These filters contain two capacitors and one inductor. The inductor is a ferrite bead or an ordinary coil as the inductor in the SFP filter. The power filter capacitor is composed of more capacitors with a different capacity. Capacitors with higher capacity have usually a higher ESR and are used to filter lower frequencies. The capacitors with lower capacity have a lower ESR and are more suitable to filter higher frequencies.

### 2.3.3 Power Distribution of the CPU Subsystem

On contrary to the ADC Subsystem, the CPU Subsystem power supplies for analogue and digital components are not separated. The power distribution of the CPU Subsystem is shown in the block diagram in Fig. 2.31. The CPU Subsystem is supplied by two different power voltages of 3.3 V and 12 V. The 3.3 V power voltage is provided by the Cyclone III Development Board through the HSMC connector. It powers the TTC receiver and the additional linear 2.5 V regulator. The 2.5 V power voltage supplies the clock input buffer, the serial number, and the user buttons.

The CPU Subsystem is provided with the 12 V power voltage either by the FPGA board or by a dedicated external connection. These two power sources are switched manually. The FPGA board should be powerful enough to supply the CPU

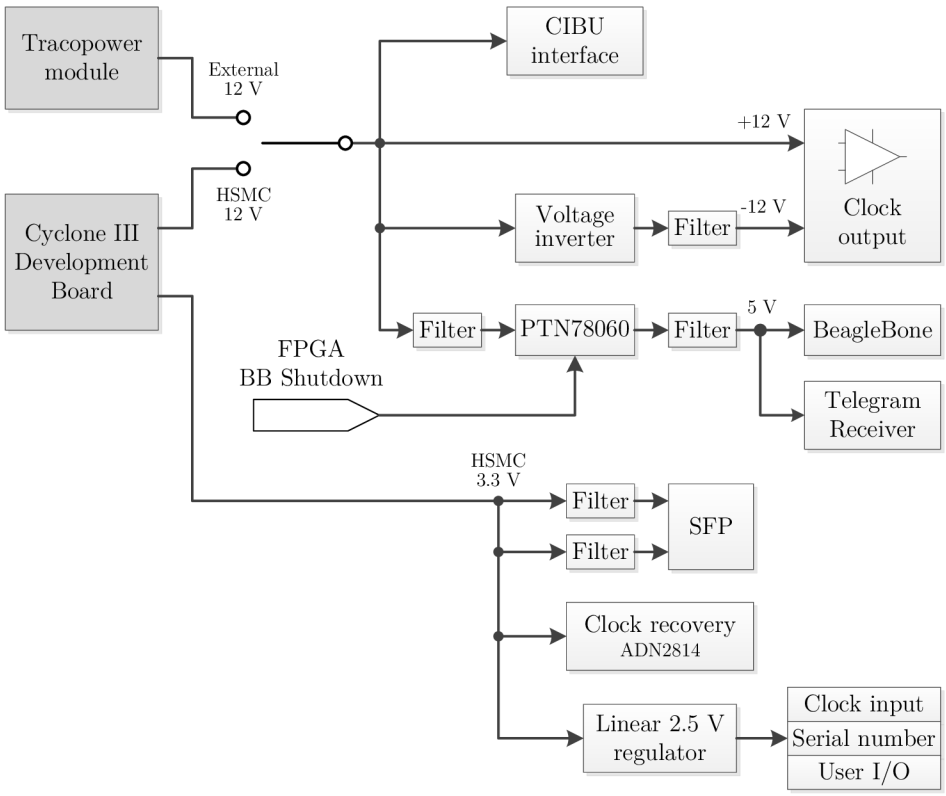



Fig. 2.31: Block diagram of the power distribution in the CPU Subsystem

Subsystem. If the CPU Subsystem consumption was excessively high the external power source would be used.

The voltage converter with switched capacitor has been designed to invert the 12 V power voltage. The negative voltage powers the output clock amplifier. The chosen voltage converter is the LTC1144. It can operate up to 18 V and requires only two external capacitors.

The 12 V power voltage has to be regulated to provide the BeagleBone with 5 V power voltage. The voltage regulator for the BeagleBone has to be equipped with a shut down input, because the FPGA needs some time to set up boot pins before the BeagleBone starts and latches them. If the BeagleBone is not powered off, it cannot latch the set boot pins due to a bug of the BeagleBone reset. The used switching mode regulator is the PTN78060 as the regulator in the ADC auxiliary power supply. The input and the output of the switching regulator are filtered to suppress interference coming from the switching. The output voltage can be turned off by the regulator inhibit input. This input is driven by the FPGA, thus the FPGA can set up boot pins prior to the BeagleBone latches them.

## 2.4 PCB Design of the FBCCM Subsystems

All electronics subsystems are implemented using technology of the printed circuit boards (PCB). The PCB layout of all the subsystems was drawn in Altium Designer.

The used substrate of all the PCBs is standard FR-4. Although the subsystems operate at RF frequencies, a RF substrate was not required.

### 2.4.1 PCB of the ADC Subsystem

The PCB for the ADC Subsystem has six layers. Its complete stack is depicted in Fig. 2.32.

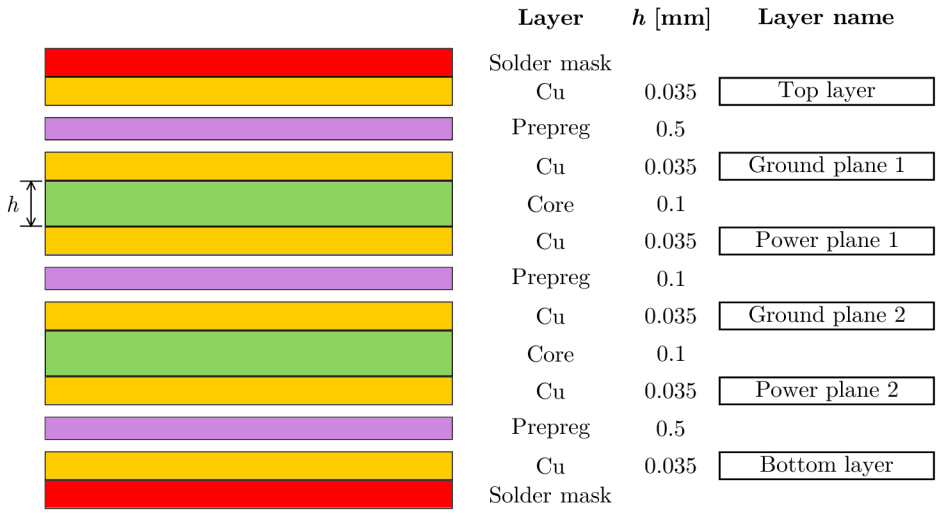



Fig. 2.32: PCB stack for the ADC Subsystem

Top and the bottom layers, where all the components are placed, are used for the general routing. In addition the PCB stack contains two power planes and two ground planes for the electric power distribution, because the ADC requires three different power voltages. Without the additional power planes, the PCB would negatively influence the system characteristics.

The power plane close to the ground planes represents a high-quality capacitor with low impedance, which is perfectly suitable for filtration of high frequencies. The capacity of a square centimetre of these planes is

$$C_{P1} = \varepsilon_0 \cdot \varepsilon_r \cdot \frac{2S}{h} = 8.854 \cdot 10^{-12} \cdot 4.8 \cdot \frac{2 \cdot 10^{-4}}{0.1 \cdot 10^{-3}} = 85.0\,\mathrm{pF} \tag{2.17}$$

for the power plane 1, and

$$C_{P2} = \varepsilon_0 \cdot \varepsilon_r \cdot \frac{S}{h} = 8.854 \cdot 10^{-12} \cdot 4.8 \cdot \frac{10^{-4}}{0.1 \cdot 10^{-3}} = 42.5\,\mathrm{pF} \tag{2.18}$$

for the power plane 2. In the equations the $\varepsilon_0$ is the vacuum permittivity, the $\varepsilon_r$ is the relative permittivity of the FR-4 substrate [23], the $h$ is the dielectric thickness, and the $S$ is the power plane surface. The surface for the power plane 1 is multiplied twice, because this plane is surrounded by two ground planes.

All the analogue and digital components in the ADC Subsystem share the common ground. The analogue and digital grounds are separated by the geometry of the PCB layout. Furthermore the most sensitive part of the ADC Subsystem, the RF Front End, is protected by a copper shielding box against electromagnetic interference.

The pictures of the implemented ADC Subsystem module can be found in Appendix A.2.1.

### 2.4.2 PCB of the CPU Subsystem

The dimensions and shape of the PCB for the CPU Subsystem is defined by the geometry of the FBCCM device. The switching power regulator of the CPU Subsystem is placed in a copper shielding box to minimise electromagnetic emission.

The pictures of the implemented CPU Subsystem module can be found in Appendix A.2.2.

## 2.5 Mechanical Design

The FBCCM device will be installed in the 19" rack in the LHC tunnel. Therefore the FBCCM box had to be selected from standard series. The box width is constant. The depth and height could be chosen in certain steps. The ADC Subsystem module is attached to the front panel. Contrarily the CPU Subsystem module is aligned with the back panel. At the same time both modules are plugged into the central Cyclone III Development Board. Thus the dimension of the entire set has to equal the depth of the selected box.

The electronic modules dimensions influence the mechanical design. Equivalently the mechanics influence the PCB layout of the electronic modules. The layout of all the PCBs and the mechanical design were performed simultaneously to meet the same dimensions. The FBCCM mechanical model has been drawn in Autodesk Inventor. The designed electronic modules were imported to the mechanical design. The 3D models of all components provided by the CERN Altium libraries were advantageously used for the import.

During the mechanical design various aspect were solved. The total length of three electronic modules assembled together has to precisely fit into the box. Any imperfection in the PCB dimensions can cause exceeding stress in the fragile electronics and break them. The total length of the affected PCBs is designed to be by 0.5 mm less than the box depth. During the device assemblage the SMA connectors of the ADC Subsystem are tighten to the front panel. The arisen gap between the

back panel and the CPU Subsystem module ensures no tension in the electronic modules. This gap is compensated by soldering the SMA connectors of the CPU Subsystem in the constructed box tighten to the back panel. The precise positioning of all the modules is ensured by lengthwise mounting slots in the bottom panel.

The mechanical model of the device is depicted in figures in Appendix A.3. The pictures of the real assembled FBCCM device can be found in Appendix A.4.

# 3 FPGA FIRMWARE

The FPGA firmware implements all data processing except the Ethernet communication interface. It is completely described in the VHDL and compiled by Altera's tools. The entities described in this chapter have some common characteristics. All their test benches are assertion-based, thus they automatically verify the desired functionality. The test benches are only behavioural; they test RTL models of the entities without the real timing information. The sequential processes are synchronised with the rising edge of the clock, if not stated otherwise.

## 3.1 Top Entity

The FPGA firmware of the FBCCM has a hierarchical structure, because the flat structure of such complex firmware would not be acceptable.

Fig. 3.1 presents the top entity of the FBCCM. The depicted block diagram is simplified; the most input and output ports and some less important entities providing auxiliary functions are omitted. The depicted arrows represent the data flow between the entities. In reality the entities are interconnected by wide buses with many signals in both directions.

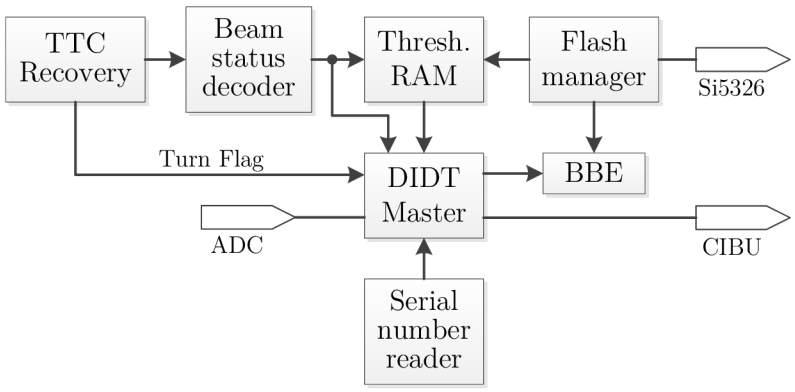



Fig. 3.1: Block diagram of the FPGA firmware

The central entity called DIDT Master acquires the signal from the ADC and processes it. When the DIDT Master evaluates losses over the predefined thresholds, it requests the beam dump. The thresholds for the DIDT Master are loaded by the flash manager when the device starts and they are stored in the threshold RAM. The threshold RAM stores 32 vectors of six thresholds. The vector passed to the DIDT Master is selected based on the actual beam momentum received by the beam status decoder. The beam status decoder receives messages from the TTC Recovery. When the TTC signal is not available, the GMT telegram is used instead of it. Besides the messages the TTC Recovery recovers the bunch clock for the ADC Subsystem PLL and the turn flag for the DIDT Master.

The PLL Si5326 is a configurable component which registers have to be set for the desired functionality. These registers are loaded once after the start of the device by the flash manager using the SPI bus. When the flash manager has loaded the thresholds and the Si5326 registers, it provides the BeagleBone Entity (BBE) with the flash data of the BeagleBone software.

In addition the serial number reader reads out and latches the serial numbers of the connected ADC and CPU subsystems. These numbers are sent inside the statistic messages generated by the DIDT Master.

## 3.2 DIDT Master

The DIDT Master is the only entity in the design performing the digital processing of the FBCT signal. Its principal algorithm has been already described in the section 1.6.1 The First Prototype. Although the described essential signal processing remains unmodified, the implementation of the DIDT Master in the final version of the FBCCM is specific due to the new interface to the BeagleBone.

The new implementation is depicted in Fig. 3.2. The entity of the moving average is instantiated six times using six different windows. The moving average output is differentiated to obtain a signal proportional to the losses. Subsequently the differential signal is compared with the predefined threshold. All this processing is performed individually for all the windows. Each window uses a different threshold. Therefore six times 64-bit thresholds are provided by the top entity.

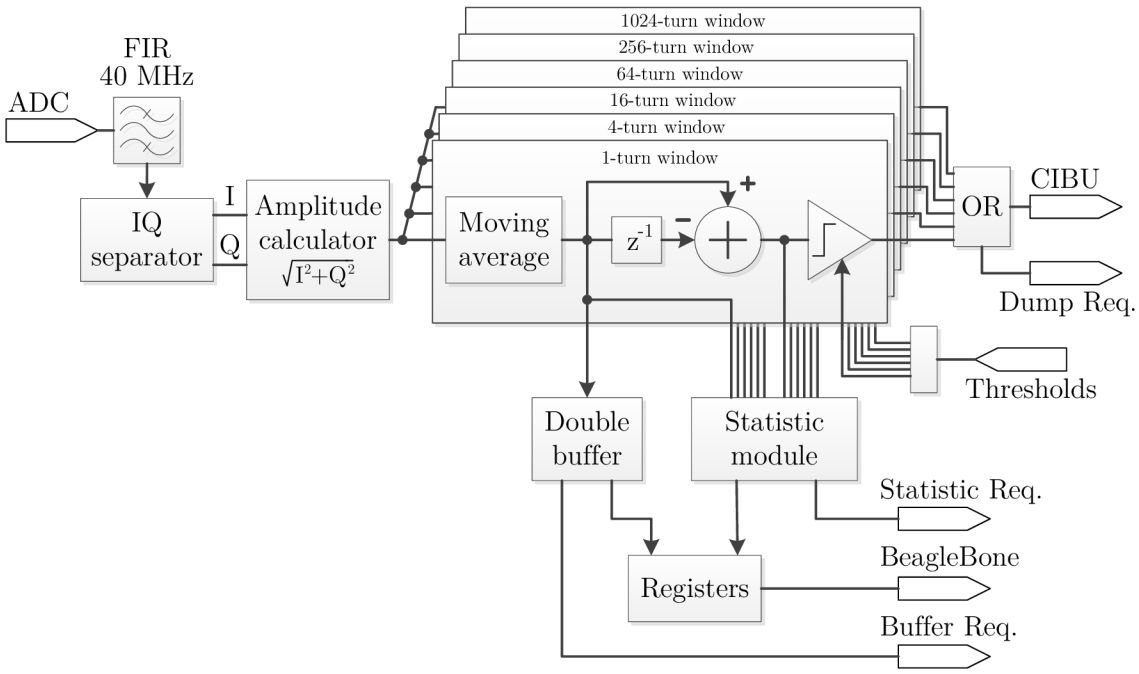



Fig. 3.2: Block diagram of the DIDT Master

The output of the 1-turn moving average is stored into a double buffer. The double buffer uses a time multiplex to store and provide the data. One buffer provides saved data when the other one is storing the averages. When the buffer storing data is filled, the storing and providing buffers are exchanged. This procedure ensures that the saved data is not corrupted by storing the new data. When the buffer is filled, the BeagleBone has to read it and send it to all connected clients. Besides the buffer data, the BeagleBone sends statistically processed data of averages and differences of all windows. The statistic processing is performed in a statistic module which evaluates the maximum, the minimum and the average of all its input signals over ten second intervals. At the end of the interval, the statistic values are sent and re-captured from the data streams again.

The BeagleBone sends the data when it is requested for it. The requests are generated by the double buffer and the statistic module. Additionally the BeagleBone sends the dump message triggered when the beam dump occurs.

The first version of the moving average is not synchronised with the turn flag, but the DIDT Master is newly required to integrate the input signal synchronously with the beam revolutions. Therefore the new moving average entity had to be implemented.

### 3.2.1 FIFO Buffer

The moving average accumulates samples of the input signal over a specified window length. Its instant output is given

$$y_i = \frac{1}{N} \sum_{i-N+1}^{i} x_j \tag{3.1}$$

where $N$ is the length of the moving window and $x_j$ is an input sample. This definition is not suitable for a parallel processing in the FPGA. The recursive definition of the moving average is used for its implementation. The multiplication factor $1/N$ is not implemented in the FPGA, as it is not necessary.

$$y_i = \frac{1}{N}(y_{i-1} + x_i - x_{i-N}) \tag{3.2}$$

The equation defines the moving average as a sum of the former sum, the new sample of the input signal, and the negation of the last window sample. Therefore the moving average has to memorise the sequence of the input samples. The sequence length equals the moving average window length.

The sequence is memorised in a FIFO buffer which is implemented in the pure VHDL. The Altera's IP core buffer is not used due to its platform dependency. It is a task for the compilation tools to decide the mapping of the user buffer into the FPGA. When the FIFO buffer is large enough and properly coded, the compiler maps it in the FPGA RAM blocks. In the end there is not any difference between the user or Altera's buffer implementation.

The implemented FIFO buffer has a single-clock interface. It consists of the clock input, the data input and output and the clock enable signal. The reset signal is omitted, as the memory blocks in the Cyclone III cannot be reset. The implemented FIFO buffer is a generic entity. Its length and bit width of the input data can be arbitrarily adjusted. It is coded as a shift register of a defined length. The shift operation is performed synchronously with the clock providing that the clock enable input is asserted.

The VHDL code of the implemented FIFO buffer can be found in Appendix A.5.1.

#### FIFO Test Bench

The FIFO buffer is a basic component. Its test bench covers the variant test scenarios (test cases). The first test scenario verifies propagation of a non-zero value through the FIFO buffer (Fig. 3.3). During this test the FIFO buffer is supplied with the enable pulses of a defined frequency. The second scenario is almost the same, but the enable input is permanently asserted. These tests verify the proper reaction of the FIFO buffer to the clock enable input. The length of the FIFO is checked during these tests as well.

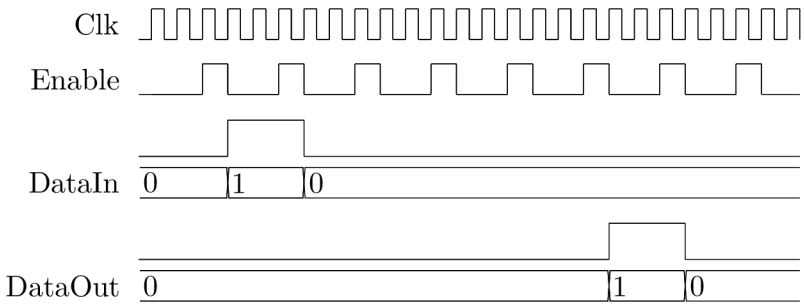



Fig. 3.3: FIFO test of the non-zero value propagation

Another test injects a ramp signal into the buffer with the last sample of the maximum value (Fig. 3.4). It verifies that the signal is not trimmed in the input bits. The other test verifies inhibition of the buffer when the enable input is not asserted. In parallel to all the tests, the output signal changes are checked to be only simultaneously with the input enable impulses.

### 3.2.2 Moving Average

The implemented moving average in addition to the calculation based on Eq. 3.2 detects the maximum and minimum of the moving average. The output samples are provided with a period specified in the number of input samples.

The block diagram of the implemented moving average is depicted in Fig. 3.5. The output enable impulses are generated in the input clock divider with frequency specified by the generic parameter. The samples of the input signal are delayed in

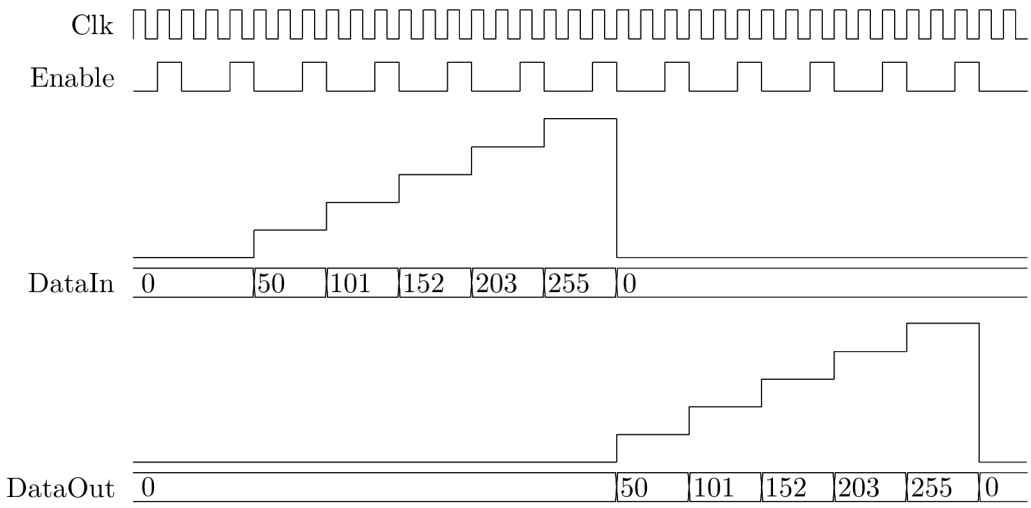



Fig. 3.4: FIFO test of the ramp propagation

the implemented FIFO buffer. The missing FIFO buffer resetting is substituted by the blanking generator, because the moving average reset is required for the purposes of the beam revolution synchronisation. If the moving average was reset without the FIFO buffer and the blanking, the FIFO obsolete samples would corrupt the output values. The blanking generator impulse switches over the multiplexer to substitute all obsolete FIFO samples by zero. This procedure ensures the safe resetting of the moving average.

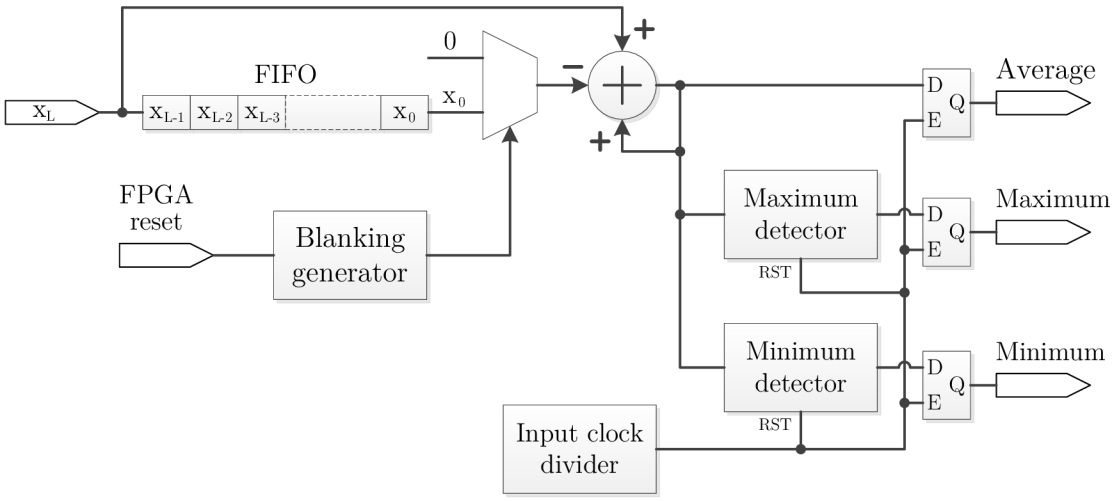



Fig. 3.5: Implementation of the moving average into the FPGA

The implemented moving average inputs are the clock, the reset signal, input data and the data enable signal. The outputs are moving average, maximum, minimum, the running average for verification purposes and the output data enable signal. Generic parameters set the length of the averaging window, the input data width, the output data decimation, and the output period. The moving average

implementation requires only unsigned data vectors, because the moving average in the FBCCM calculates the sum of the signal amplitude which is never negative.

### Test Bench of Moving Average

The test bench of the moving average is complex and more complicated than the moving average itself. It is composed of nine test scenarios to verify the main functionality. In parallel the additional tests verify the output frequency and that the average is always between the minimum and maximum.

The particular test cases verify

1. the minimum and maximum values after the entity is reset,
2. the length of the blanking interval replacing the reset input,
3. the full speed input data processing,
4. the registration of the last sample,
5. the averaging window length,
6. the immunity against overflow,
7. the processing of a ramp up and down between two consecutive output samples,
8. the processing of a long ramp up,
9. and a long ramp down.

The entity under test provides the output values after two lengths of the averaging window, thus only the last half of the signal is averaged. The moving average of the entire signal is reflected into the maximum and minimum values. All the tests, except the full speed test, provide the entity with the input samples of a defined frequency. This should be sufficient to verify reaction on the enable signal. If the entity was latching the input signal in wrong moments, the moving sum would be incorrect.

Between all the tests the window of the moving average is flushed out, consequently any test cannot influence the others. Additionally each test case is aligned between two consecutive outputs. At the end of each test, the output values can be unambiguously verified.

### Test Case 1

This test (Fig. 3.6) verifies the first maximum and minimum values after the moving average is reset. It could happen that the reset would set maximum and minimum into inappropriate starting values. This would not be observed at later output values, because the resetting of the maximum and minimum detectors by the output sample is different than the asynchronous resetting. The proper starting values for the detectors are the zero for the maximum detector and the full-scale value for the minimum detector which ensure that both detectors will latch the sum with the first input sample.

During this test the test bench provides the moving average with a constant signal. Its value should become the minimum. At the end, the maximum and the average equals the input value times the window length.

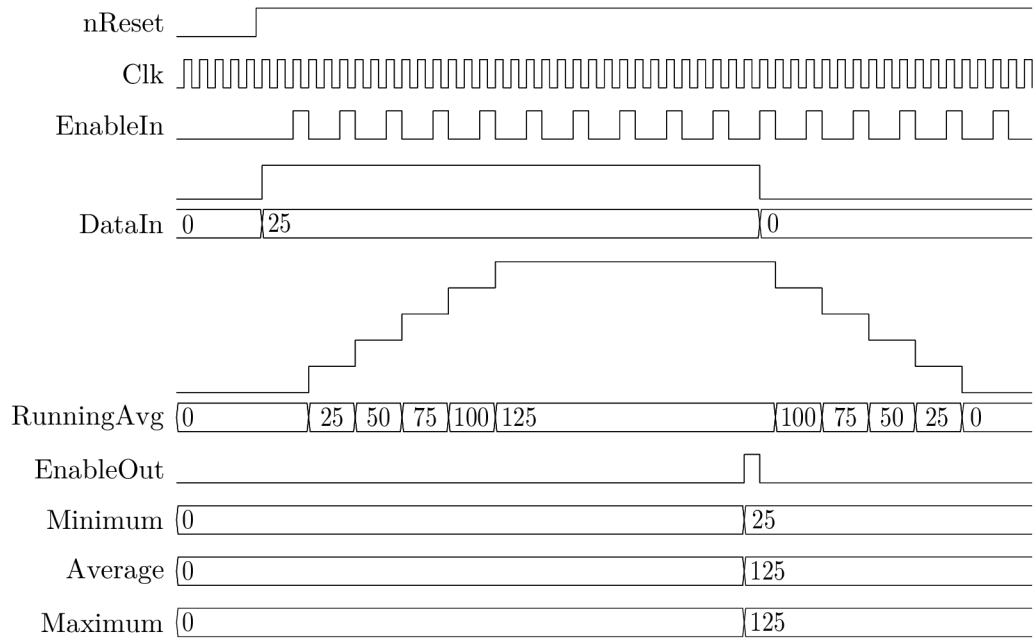



Fig. 3.6: Moving average test of the first maximum and minimum after reset

### Test Case 2

The second test case (Fig. 3.7) looks after the length of the FIFO blanking interval after the entity resetting. The blanking of the FIFO has to be performed in the defined moment to cover all the invalid samples in the FIFO and only the invalid samples. If the blanking interval was incorrect, the invalid sample from the FIFO could be subtracted from the running sum or the valid sample could be missed. In both cases all the output values after resetting would be incorrect.

The test case fulfils the buffer with a number $x$. The moving average entity is reset and the first input sample is set to the value $y \neq x$. All the other following input samples are zero. The output values have to correspond only to the value $y$ and may not be influenced by the value $x$.

### Test Case 3

The input clock enable is forced to be permanently asserted during this test. A predefined value is set as the first sample and the waveform of the running sum is compared with the expected sequence.

### Test Case 4

The last sample test (Fig. 3.8) verifies the registration of the last input sample. The input sample right before the output sample could be calculated only into the average and not properly registered by the maximum and minimum detectors.

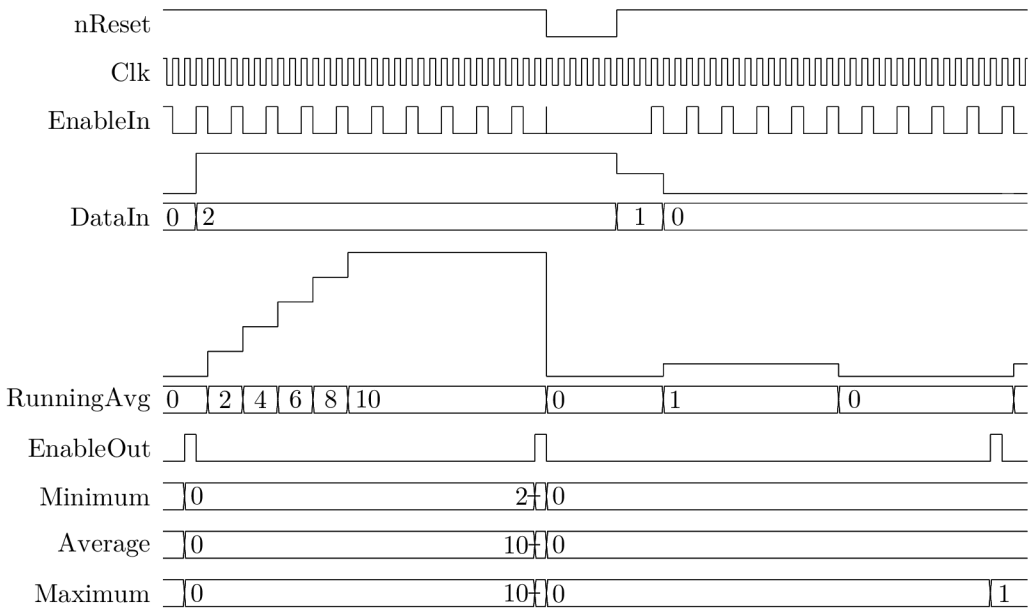



Fig. 3.7: Moving average test of the blanking interval

During the test the entire window is filled with zeros. A non-zero value is sent to the moving average as the last one. When the output is provided, the average and maximum values have to equal the last input sample and the minimum has to be zero.

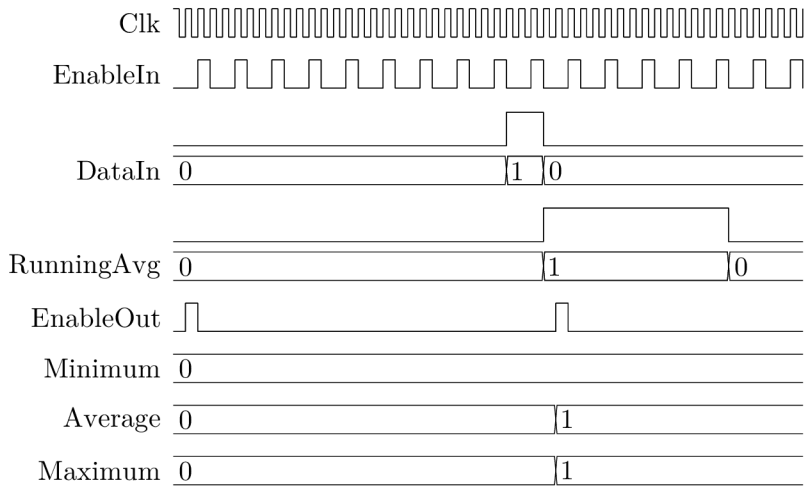



Fig. 3.8: Moving average test of the last sample registration

### Test Case 5

The length of the moving average window is verified by this test (Fig. 3.9). The input samples during the whole test have a constant value. The averaging window is filled with the input samples. The running sum makes a linear ramp up until the window is filled. From that moment the running sum is constant, because samples

leaving the FIFO buffer compensate the input value. At the end the output average and the maximum equal the input value times the length of the window and the minimum output is exactly the input value.

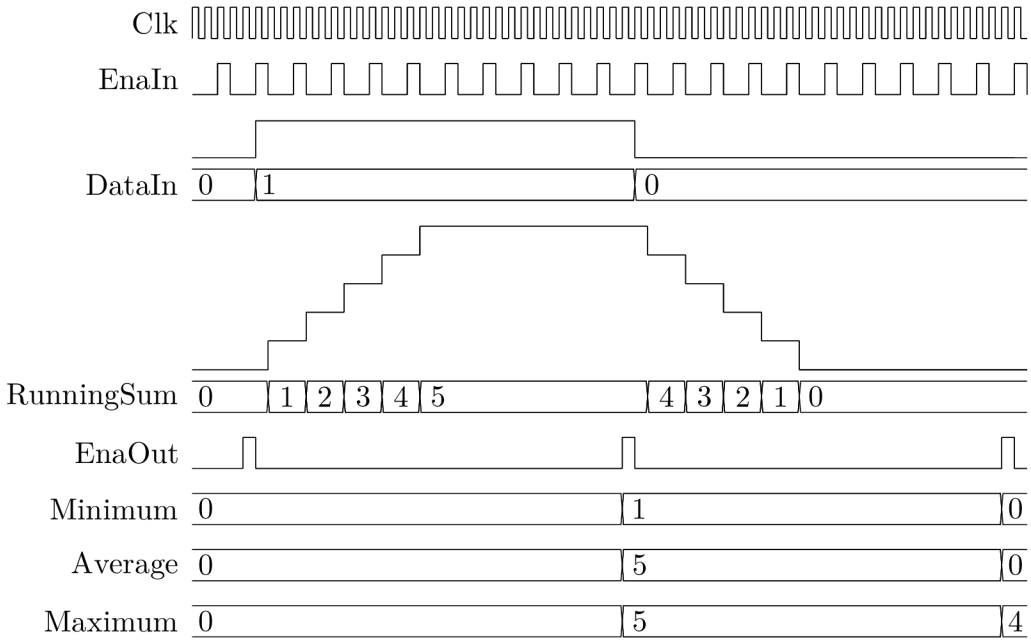



Fig. 3.9: Moving average test of the window length

### Test Case 6

The overflow test is same like the window length test. The only change is that the input value is $2^w$ where $w$ is the bit width of the input data.

### Test Case 7

This test scenario is not aimed at any particular feature of the moving average. It verifies processing of the changing input signal (Fig. 3.10). The input signal for this test is a linear function divided in two intervals. In the first interval the function is rising up to the maximum value and in the other it is falling back to the zero. The generated ramp up and down is exactly aligned between two consecutive outputs, thus the entire ramp down fits exactly into the moving window at the last position.

The running sum is continuously compared to the quadratic function waveform which is generated by the summing of the input linear function.

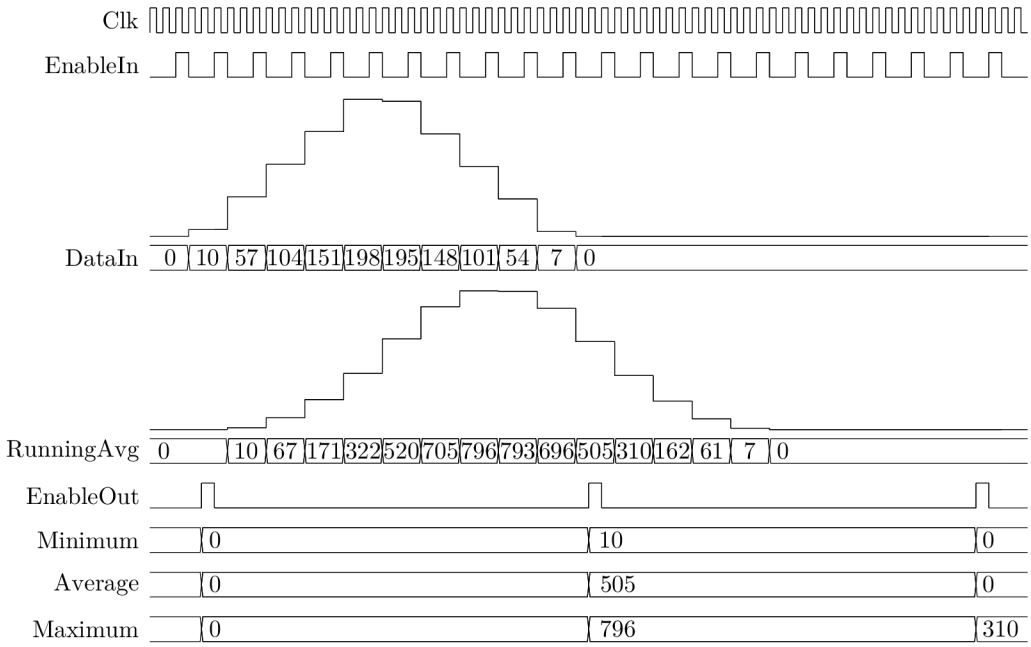



Fig. 3.10: Moving average test of the ramp up and down

### Test Cases 8 and 9

These test cases inject the linear function into the moving average. In the first test case the entity calculates the moving average of the rising function (Fig. 3.11), in the other the function is falling (Fig. 3.12). The functions in both cases are spread over the entire interval between two output samples.

Some performed tests are redundant. The overflow test verifies also the length of the window as well. The other similar relations between tests can be found. Although only one test could verify the whole functionality, it would be extremely complicated. Additionally the test cases are independent, thus a test case can be changed easily. The individual tests help to define potential problems of the entity code.

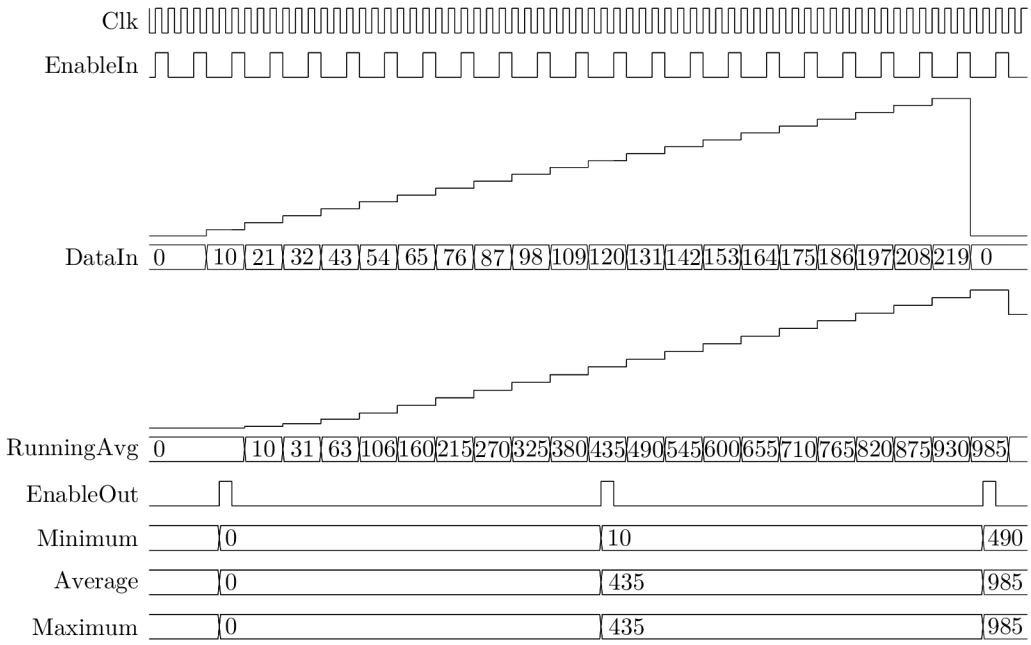



Fig. 3.11: Moving average test of the long ramp up

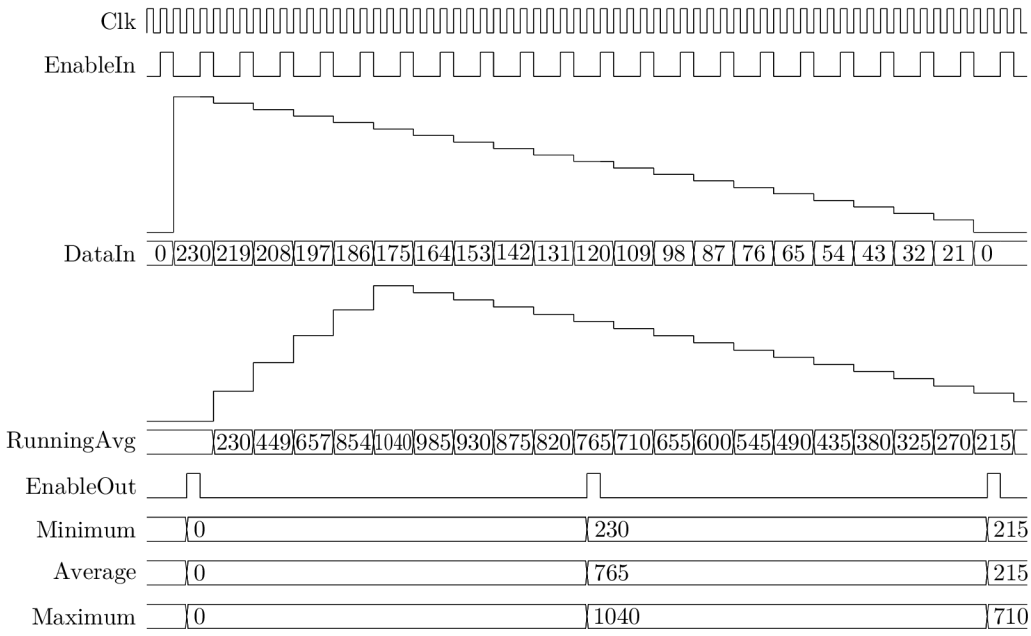



Fig. 3.12: Moving average test of the long ramp down

### 3.2.3 Turn Flag Synchronisation

The moving average is reset by the DIDT Master synchronously with the turn flag which is generated in the TTC Recovery. The DIDT Master works in a different clock domain than the TTC Recovery. Its clock is derived from the 160 MHz sampling clock provided by the ADC. The frequencies of the TTC Recovery clock and the turn flag destination clock are the same, but still the clock domains are asynchronous. The real frequency of the destination clock can be slightly lower or higher than the frequency of the source clock.

The turn flag is a single clock pulse which repetition frequency equals the revolution frequency of $\approx 11.3$ kHz For its synchronisation the common circuit depicted in Fig. 3.13 is used. Example waveforms of this circuit are shown in Fig. 3.14.

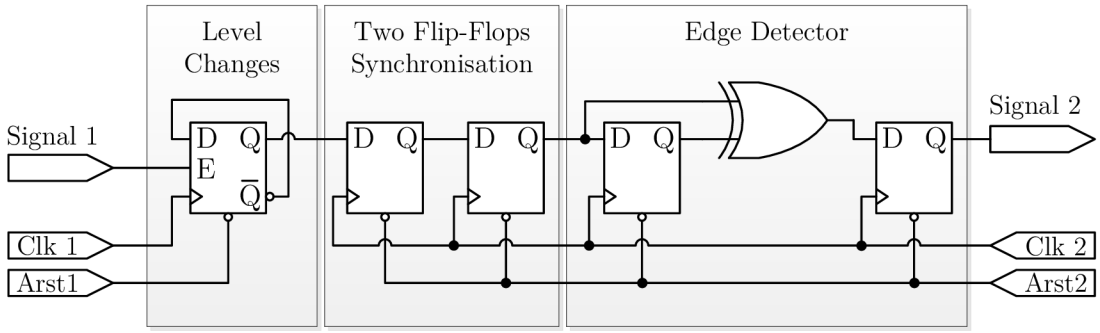



Fig. 3.13: Implementation of the turn flag synchronisation circuit

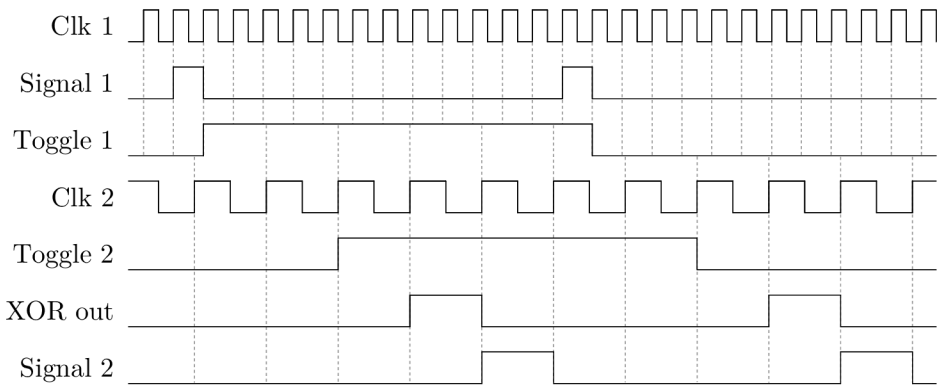



Fig. 3.14: Example waveforms of the implemented synchronisation circuit

The trick is to convert the input impulses into the level changes in the source clock domain. The alternating signal (Toggle 1) can be easily synchronised by two flip-flops to the destination domain (Toggle 2). The only limitation of this synchronisation is the minimum delay between the input impulses. The delay has to be longer than the period of the destination clock. Otherwise the level change caused by two fast consecutive impulses could be missed by the two flip-flops; consequently

both impulses would be missed out. At the end the impulse character is recovered from the synchronised signal by the edge detector.

In addition the synchronisation circuit is equipped with two reset inputs. It is necessary to emphasise that either both reset inputs have to be connected to the main reset or none of them. If only one reset input was connected, the synchronisation circuit could possibly generate a false impulse after resetting.

The main advantage of this synchronisation circuit is its independence from the source and destination clock frequency ratio.

#### Test Bench of the Synchronisation Circuit

The functionality with the both clock configurations is verified by the test bench. At the first time the frequency of the destination clock is lower than the frequency of the source clock. The other time it is vice versa. The test bench generates two consecutive input impulses and verifies the delay of the output impulses. The output impulse has to take exactly one clock period.

The VHDL code of the implemented turn flag synchronisation circuit can be found in Appendix A.5.2.

## 3.3 BeagleBone Interface

The FPGA firmware has to implement an interface to provide the BeagleBone with the data generated by the DIDT Master. Additionally this interface has to initialise the BeagleBone, because the SD is not used to store the BeagleBone application. The BeagleBone bootloader and the main application are loaded from the flash of the Cyclone III Development Board, thus all the BeagleBone software can be upgraded remotely by the Altera Ethernet Blaster.

### 3.3.1 BeagleBone Initialisation

The BeagleBone initialisation process starts by the execution of the first bootloader stored in the processor ROM. This bootloader sets up PLLs and clock domains of the processor. It reads out boot pins defining the start-up options which include settings of the boot sequence. During the boot sequence the processor configures a controller of the peripheral device and tries to locate a valid user bootloader in that device. If the user bootloader is not found, the processor continues with the next device in the boot sequence. The devices usable in the boot sequence are defined in [24]:

- the NOR memory or the other execute-in-place (XIP) device,
- the MMC or SD card,
- the SPI flash memory,

- the NAND flash memory with the geometry read from an I2C EEPROM,
- the UART,
- the Ethernet MAC,
- or the USB.

The SPI, USB, Ethernet MAC, and MMC or SD card are not accessible through expansion connectors, thus they cannot be used for the loading by the FPGA. The throughput of the UART is insufficient for the purpose of the application loading. The interface of a NAND flash memory could be used, but the FPGA would have to emulate its geometry and paging. The last remaining option is the XIP device. The XIP device allows the processor to execute a user bootloader directly from the device, thus the processor does not need to copy the bootloader into the internal RAM. This interface provides the sufficient data throughput and can be emulated by the FPGA.

If the XIP device is in the boot sequence, the processor configures the General Purpose Memory Controller (GPMC) for this device. The GPMC is configured for communication in asynchronous mode with either 8 bit or 16 bit data bus. This depends on the start-up options. The XIP device is expected to be connected to the defined chip select. If the XIP device contains a user bootloader, the processor starts executing it.

The advantage of using the XIP device for the bootloader loading is that the same configuration of the GPMC can be used for further data transfers, i.e. the main application loading and DIDT Master data reading. In consequence of that the same FPGA entity can advantageously transfer all data to the BeagleBone.

### 3.3.2 BeagleBone Entity

The BeagleBone Entity provides a data link between the DIDT Master and the BeagleBone. Additionally it provides the BeagleBone with the flash data during its initialisation process. The block diagram of the BBE implemented into the FPGA is presented in Fig. 3.15.

The BBE has to set the boot pins for the XIP device booting, because in the default configuration the bootloader is loaded from the SD card. Subsequently the BBE turns on the power supply of the BeagleBone. When the processor is powered up, it latches the boot pins. At the moment the BBE switches them on the FPGA side to the input mode as they form the address of the GPMC interface. At the end of the sequence, the processor reset is released and the booting procedure can start. During the booting procedure the processor reads the flash data which it is provided with by the GPMC Slave entity.

In addition the GPMC Slave provides the BeagleBone with the DIDT Master data. These data transfers are triggered by the DIDT Master single clock requests. These are too short to be latched directly by the BeagleBone. Therefore the impulse generator stretches them to generate interrupt requests for the BeagleBone. Due to the similar reason, the reset circuit provides the BeagleBone with the reset signal.

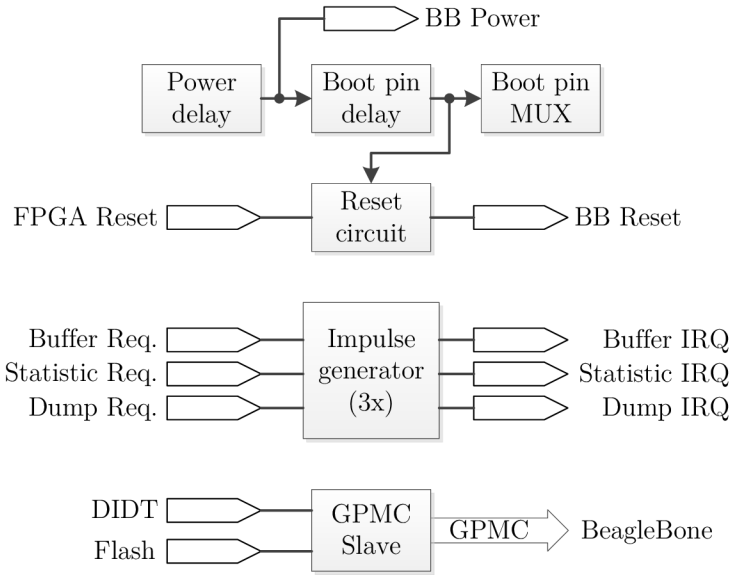



Fig. 3.15: Block diagram of the BeagleBone entity

### 3.3.3 GPMC Slave

The GPMC Slave is a communication interface transferring the data between the BeagleBone and the DIDT Master or the flash memory. The BeagleBone GPMC expects a 16 bit data width NOR memory and is set up to multiplex its data and address buses. The separated buses would be more suitable for the FPGA implementation, but not all interface pins are available at the expansion connectors. The GPMC monitors the wait pin to determine if the valid data is on the bus. The GPMC Slave uses this functionality to minimise various latencies caused by requesting the data from different sources. The data latency from the flash memory is considerably higher than from the DIDT Master. The GPMC Slave is not required to fulfil any strict timing constrains due to the wait pin monitoring.

The GPMC Slave allows the BeagleBone to perform only the read operations. All the write requests are ignored due to the security reasons.

Fig. 3.16 shows typical single read access. The data transfers are asynchronous and the depicted clock (GPMC_FCLK) is a processor internal signal. Before the read cycle start all the signals are set to their inactive levels. The read cycle starts with the assertion of the chip select (nCS) signal and a valid address. The address valid signal (nADV) is asserted later. The connected device latches the address and asserts the wait pin. The GPMC switches over the shared data bus into the input mode and asserts output enable signal (nOE). The connected device drives requested data on the bus and releases the wait signal. The GPMC latches the requested data with the rising edge of the wait signal. The read cycle ends with the release of the nOE and nCS signals.

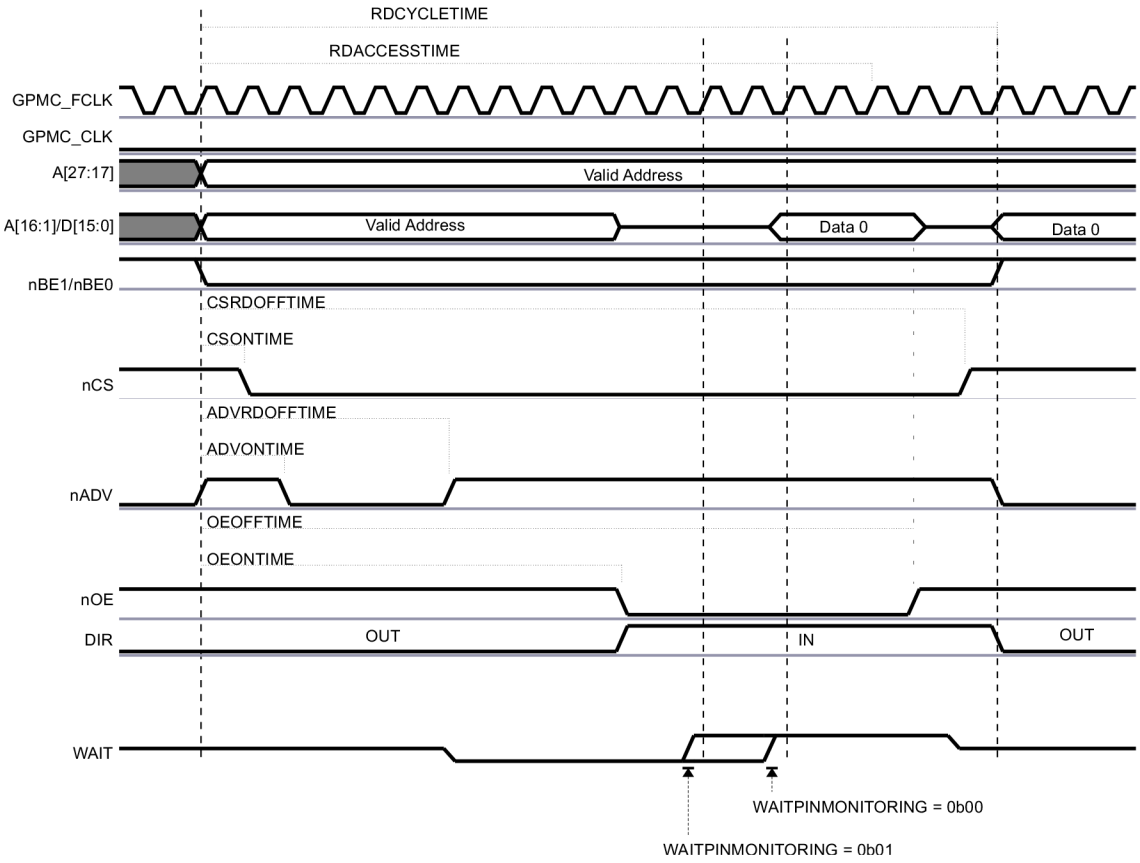



Fig. 3.16: Asynchronous single read access with wait pin monitoring [24]

### GPMC Slave Implementation

The described communication procedure is implemented by the GPMC Slave into the FPGA as a finite state machine (FSM). The state diagram of the GPMC Slave is depicted in Fig. 3.17. For the sake of the simplicity, the state diagram shows only the important changes of output signal instead of all the output assignments.

The implemented FSM is Mealy type. Its state diagram corresponds to the convention shown in Fig. 3.18. The input conditions and output assignments are related to the transitions. This reflects immediate output assignments depending on a fulfilled input condition.

After the reset or the start up, the FSM starts in the state *idle* where it waits for a valid address (nADV = 0) and an active chip select (nCS = 0). When the valid address is latched, the FSM continues either to the *didt_req* or *flash_ready* state depending on the highest address bit. In the *didt_req* state the FSM generates a read request of the DIDT data and waits one clock cycle in *didt_wait* for the requested data. This path takes constantly three clock cycles.

If the FSM reads data from the flash memory, it needs to wait for the flash memory readiness in the *flash_ready* state. Subsequently it generates read request and waits for the flash data in the *flash_data* state. The delay of the flash data

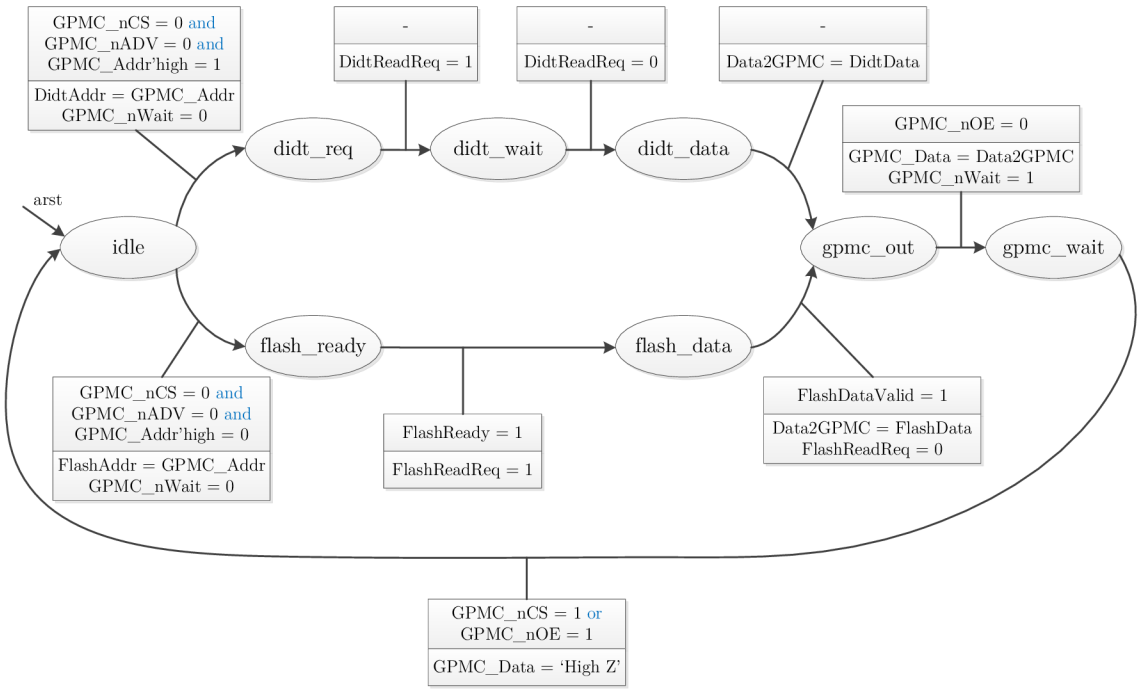



Fig. 3.17: State diagram of the GPMC Slave

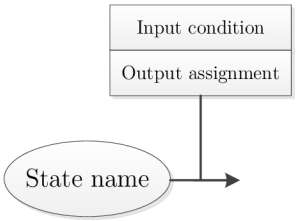



Fig. 3.18: Mealy FSM state diagram convention

arrival depends on the flash memory access time.

Both paths continue in the *gpmc_out* state by waiting for the output enable (nOE) flag. When the output is enabled, the GPMC Slave drives data on the bus and releases the wait signal. Afterwards it waits in the *gpmc_out* state for the release of the nOE or nCS.

### GPMC Slave Test

The GPMC Slave is implemented to comply with the GPMC description in [24] which does not involve all possible situations. The behavioural model of the GPMC is not available, thus the implemented GPMC Slave and its test bench can cover only the described behaviour.

During the real test, the BeagleBone application reads out the entire GPMC

address space. The GPMC Slave in the FPGA returns the negated address as the data for all read requests. The BeagleBone afterwards verifies the data against the implemented equation. If all the data is compliant with the expectation, the test is successfully passed.

## 3.4 Flash Data Upgrade

The data provided by the flash to the BeagleBone and the others have to be written into the flash memory by the Altera Ethernet Blaster (EBL). The flash memory is not connected into the JTAG chain, thus the EBL programmer cannot access it directly. The Altera's example firmware with the NIOS processor is loaded into the FPGA prior to upgrade. The NIOS processor provides a link between the EBL and the flash memory. Subsequently the Altera's application is used by any network computer to load the desired data region into the flash memory. The flash memory is divided into regions based on the their data content which are

1. the FPGA firmware,
2. the Bootloader,
3. the BeagleBone application,
4. the Si5326 register initial values,
5. and the DIDT thresholds.

At the end of the upgrade, the EBL loads the FBCCM firmware into the FPGA.

This complex procedure is executed by a make file, thus the upgrade of the entire flash can be performed by a single command.

# 4 BEAGLEBONE SOFTWARE

The microprocessor module BeagleBone implemented in the FBCCM is not driven by any operating system. The solution of the stand-alone application has been selected, because of its easier maintenance and the CERN security policy.

The source codes of the BeagleBone software are written in the programming language C. They are compiled by the ARM GCC tool chain into the processor binary executable code.

## 4.1 BeagleBone Bootloader

The binary code of the main application is loaded into the BeagleBone by an application called the bootloader. The processor executes the bootloader directly from a XIP memory emulated by the FPGA. The XIP memory is implemented as a read-only device. Consequently it stores only the data constants and the bootloader binary instructions, but it cannot store the bootloader data variables. The Bootloader data variables and the constant data are separated by the compiler into individual program data sections. These sections are organised and mapped in the physical memories during the linking process by a linker. The linker had to be ordered to link all data variables into the processor internal SRAM which ensures that these variables are writable.

The SRAM memory is not permanent and has to be initialised after the processor starts. Its initialisation is performed by a start-up code. The start-up code is a small routine written in the assembly language which initialises the stack pointers and the zero data variables in the '.bss' section. It is added by the linker before the first instruction of the bootloader main function. For the bootloader, the start-up code was extended to initialise the non-zero data variables in the '.data' section as well. Their initial values are saved by the linker in the XIP memory. The start-up code copies them from the XIP memory to the corresponding locations in the internal SRAM. Consequently the bootloader main function finds all the variables containing the correct initial values.

The bootloader instructions have to be read from the emulated XIP memory before they are executed. The instructions are read from the emulated XIP memory with high latency. Consequently the bootloader execution is extremely slow. To speed up the bootloader, the instruction cache memory was enabled. The bootloader instructions are stored in the fast cache memory, thus instructions of loops are loaded only once from the external XIP memory. Subsequently they are executed from the cache memory which causes faster bootloader execution.

The operations performed by the bootloader are depicted in Fig. 4.1. The bootloader initialises the controller of the BeagleBone 256 MB Dual Data Rate (DDR) Dynamic RAM (DRAM). It continues by the GPMC reconfiguration for the further communication, because the first ROM bootloader uses the GPMC bus with the 16 bit addresses which is not sufficient for the main application loading. When the

initialisation is finished, the bootloader prompts the UART channel of the USB chip. If it receives a response, the main application is loaded from the UART. This is designed for the fast loading of the main application during its development, because it allows to boot from USB and to avoid the slow loading of the main application into the flash memory. If the bootloader has no response from UART, it loads the application from the flash memory through the GPMC. In both cases the application is loaded into the DDR memory. When the loading is finished, the processor jumps to the first instruction of the main application and starts executing it.

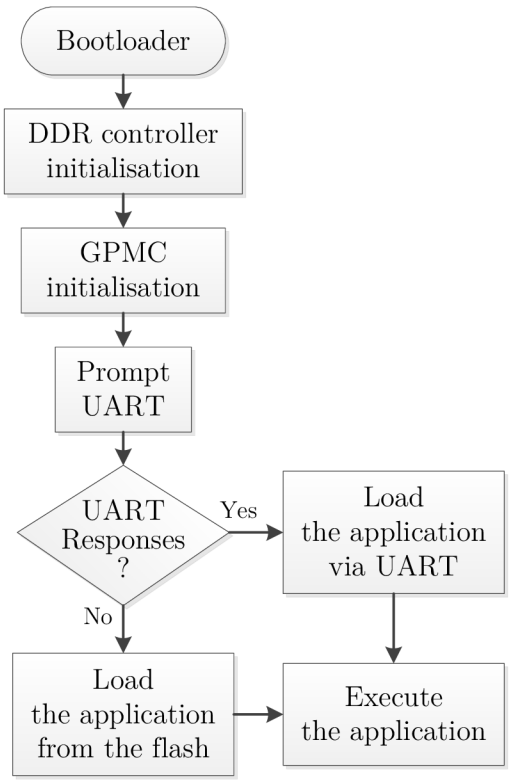



Fig. 4.1: Bootloader flow chart

## 4.2 BeagleBone Main Application

The main application provides the FBCCM with a communication interface. It manages client connections and sends them the DIDT Master data. The main application is divided by the function relevance into the programme modules (Fig. 4.2). These modules are encapsulated to get closer to the object-oriented programming. It is better for the program abstraction and the independent modules can be analysed more easily.

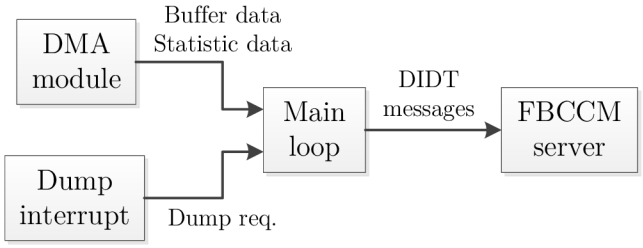



Fig. 4.2: Block diagram of the main application

### 4.2.1 DMA Module

The DIDT Master data is transferred into the BeagleBone DDR memory by the GPMC configured as it has been described in Sec. 3.3 BeagleBone Interface. The GPMC performs the data transfers in the lowest level. The processor has to generate addresses which the GPMC has to read out. If the processor needs to read a range of addresses, it has to provide the GPMC with address by address. Additionally the processor has to consecutively store the read data in its memory if the data are required to be processed as one block.

If the processor core performed this operation, it would spend a lot of time on transferring the data from the FPGA into its memory due to the data transfer latency. Therefore the direct memory access (DMA) controller is requested to transfer the data from the address range, pointing into the FPGA memory space, into the DDR memory instead of the processor core. The information about the requested transfer is written into the DMA controller's registers. They are stored in structures called the DMA descriptors.

Multiple DMA descriptors can be configured to provide the DMA controller with future orders. Additionally these descriptors can be linked, thus the DMA controller can process them without waiting for the new transfer information. Due to that the DMA controller can effectively and consecutively transfer different data.

The linked descriptors allow performing a technique called the ping-pong buffering. The first descriptor informs the DMA controller to transfer the desired data to the ping buffer. The other usually describes the same data transfer to the different pong buffer. The ping pong buffering provides the processor core with some time to process the transferred data. The processor core can process the ping buffer while the pong buffer is being transferred, and vice versa.

The main application uses a ping-pong buffering to transfer the DIDT Master data into the BeagleBone memory. The data are transferred by two channels of the DMA controller. The first channel is dedicated to the statistic data, the other to the turn data. The linked DMA descriptors of one DMA channel are shown in Fig. 4.3. The figure depicts only the basic descriptor parameters. The details can be found in [24]. The source address of the statistic DMA channel points at the beginning of the statistic registers in the FPGA. For the turn data channel, the source address is set to the start of the turn data registers.

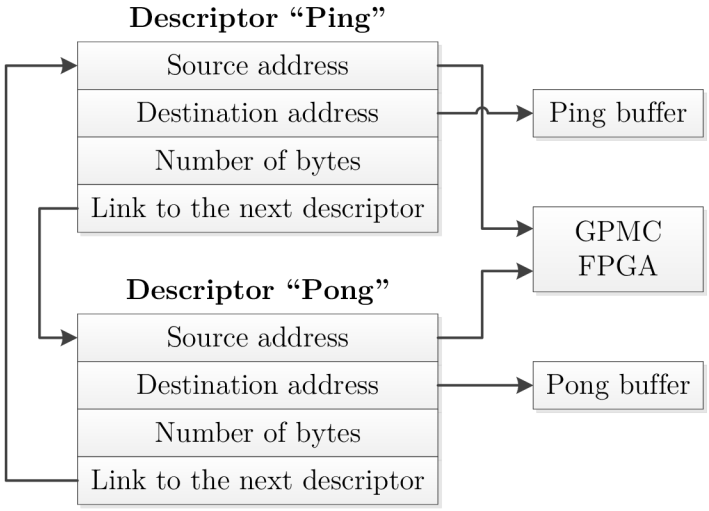



Fig. 4.3: DMA descriptors for the ping-pong buffering

The DMA channel transfers are triggered by two independent interrupt requests received from the FPGA without a processor core intervention. This ensures no excessive latency of the DMA transfers regardless the processor core utilisation. The transfer procedure is depicted in the flow chart in Fig. 4.4. When the DMA controller receives an interrupt request, it processes the current descriptor. After the descriptor transfer is finished, the DMA controller moves to the following descriptor and waits for another transfer request. The processor core is informed about the completed transfer by an interrupt generated by the DMA controller.

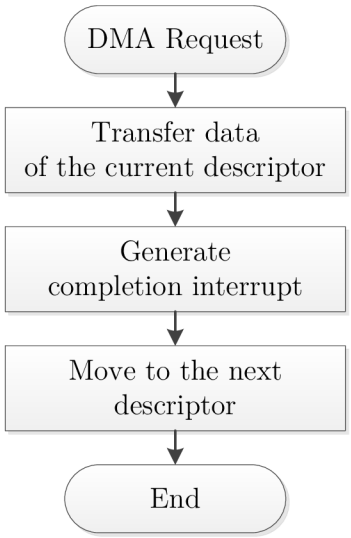



Fig. 4.4: Flow chart of the DMA interrupt processing

### 4.2.2 Main Loop

The interrupts of the DMA controller are processed by the dedicated interrupt handler. This handler asserts a flag identifying the completed descriptor. The main loop repeatedly polls these flags (Fig. 4.5). If the main loop finds out the asserted buffer flag, it determines the completed descriptor and sends the turn data message with the corresponding buffer. After the message is sent, the main loop releases the flag, thus it can be asserted again by the interrupt handler. The procedure of the statistic channel data processing is equivalent.

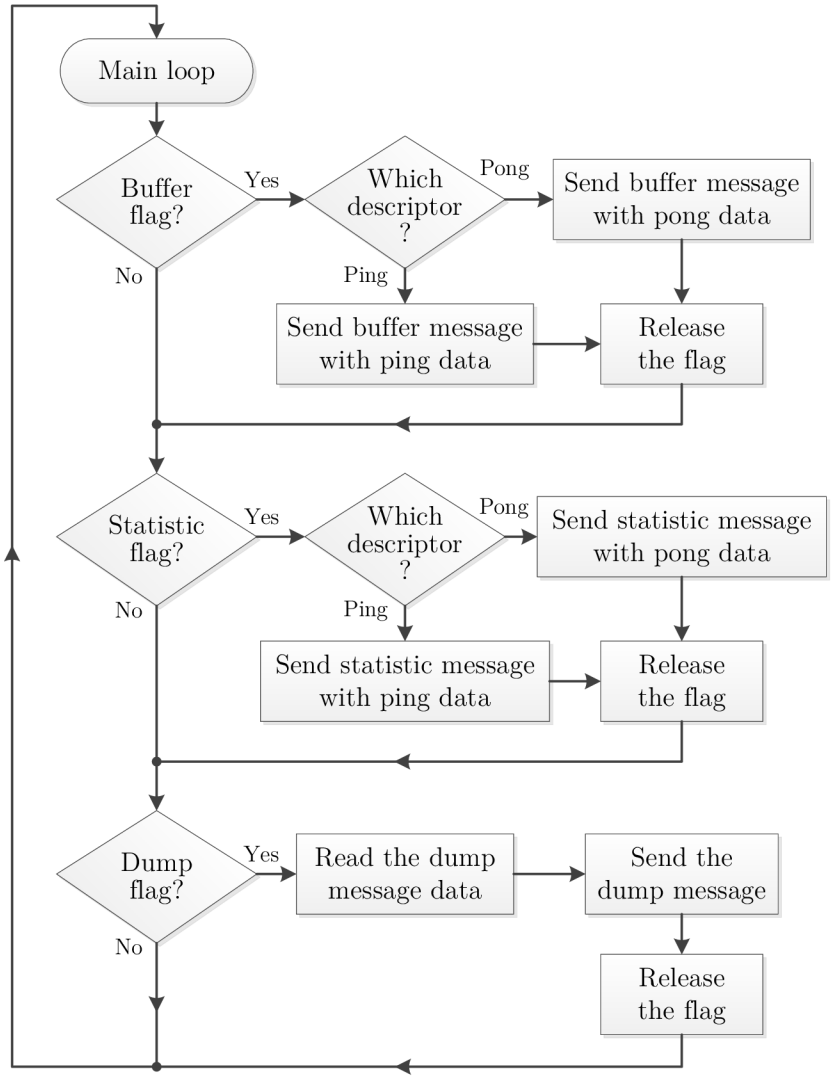



Fig. 4.5: Flow chart of the application main loop

Besides the buffer and statistic data, the main loop sends the dump messages. These messages are triggered by an ordinary interrupt. The DMA controller cannot be triggered by the third external interrupt, thus the values of the dump message are read directly by the processor core. Their reading does not limit the processor performance, because the dump messages are sent rarely and few bytes long.

When the FPGA triggers the dump interrupt, its handler asserts a dump flag. The asserted flag is registered by the main loop. Consequently the beam dump message values are read out from the FPGA registers. Subsequently the message is sent and the flag is released.

All the interrupt flags in the main loop are being read and released in the processor protected mode when all the interrupts are disabled. If the manipulation with the flags would be performed in the normal mode, the occurred interrupt in the moment of the flag reading or releasing could corrupt the flag.

### 4.2.3 FBCCM Server

All the described messages are sent to the Ethernet network by the FBCCM server. The FBCCM server is a module which allows remote clients to connect to the FBCCM. It administrates client connections and provides the main application with a function for sending data to all the connected clients.

The data is sent through the TCP. The TCP and the lower software protocols of the communication model are implemented by the lwIP stack. The lwIP stack is an open source implementation of the TCP/IP protocol suitable for microprocessor applications. The lwIP stack functions are executed inside an interrupt handler of the hardware timer independently of the main application. Therefore the lwIP functions have to be called from the main application only in the protected mode. Otherwise the interrupt handler processing would corrupt the lwIP memory. The FBCCM server has to provide the lwIP stack with call-back functions which allow the lwIP stack to perform the application layer operation. These call-back functions are executed when certain events occur. The call-back functions are registered during the FBCCM server initialisation.

The FBCCM server initialisation creates a listening socket and binds it to a defined port. This socket is used to receive connection requests from the clients. When the lwIP receives a connection request, it calls the particular call-back function. This function looks for a free client slot in the pool. The number of client slots is limited by the application, thus if no free slot is found, the client connection is denied. If a free slot is found, the requesting client is connected and bound to this slot. The bound slot carries the information about client and status of its connection. At the end, the call-back function informs the client about the successful connection.

The connected clients are required to periodically send an acknowledgement string to the FBCCM. This procedure ensures disconnection of all inactive clients and frees the client slots for another potential client. It is implemented by the FBCCM server. Each connected client has its own disconnection timer. All the client timers are increased in one second steps by a dedicated hardware timer interrupt. If the client timer reaches a defined value (currently 10 s) before the acknowledgement is received, the corresponding client is disconnected. The particular client timer is zeroed, when the FBCCM server receives an acknowledgement string from the client. This functionality of the FBCCM server is performed independently of the main loop processing.

The FBCCM server provides the main loop with the function for sending messages to all connected clients. The sending function goes through the client pool and sends the data to all the connected clients. The data is not copied by the lwIP stack into the output buffers, because it is constant for enough long time due to the implemented ping-pong buffering. The lwIP stack forms a message with all the TCP/IP headers and requests the DMA controller to send the entire message. Subsequently the processor does not need to care about sending anymore, thus this operation does not consume much processing time.

### 4.2.4 Structure of the FBCCM Messages Sent to Clients

The three types of messages are sent to clients. All these types have their specific structure which is composed of a header, a body, and footer. The message body contains different data of the FPGA registers. The data is ordered as in the FPGA address space, thus the BeagleBone can send this data without any additional processing.

The header and the footer are added to the messages by the BeagleBone. Because they are constant for the same message type, the main application fills them once during the initialisation. The headers contain a four bytes message identifier and a four bytes message length. The footer is a four bytes code, constant for all the message types.

#### Turn Data Message

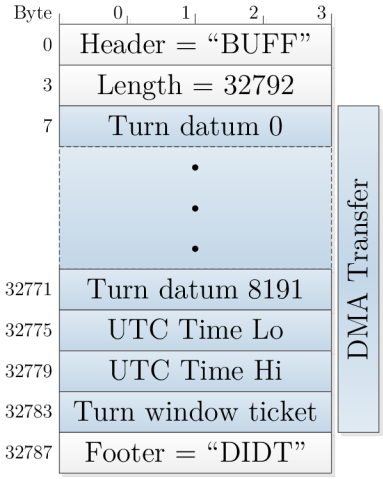



Fig. 4.6: Structure of the turn data message

The structure of the turn data message is depicted in Fig. 4.6. The message body contains 8192 consecutive samples of the one-turn moving average. These samples are followed by the 64-bit time stamp and the turn window ticket. The turn window

ticket is an incremental number which is increased by the FPGA each time the turn data is sent.

### Statistic Message

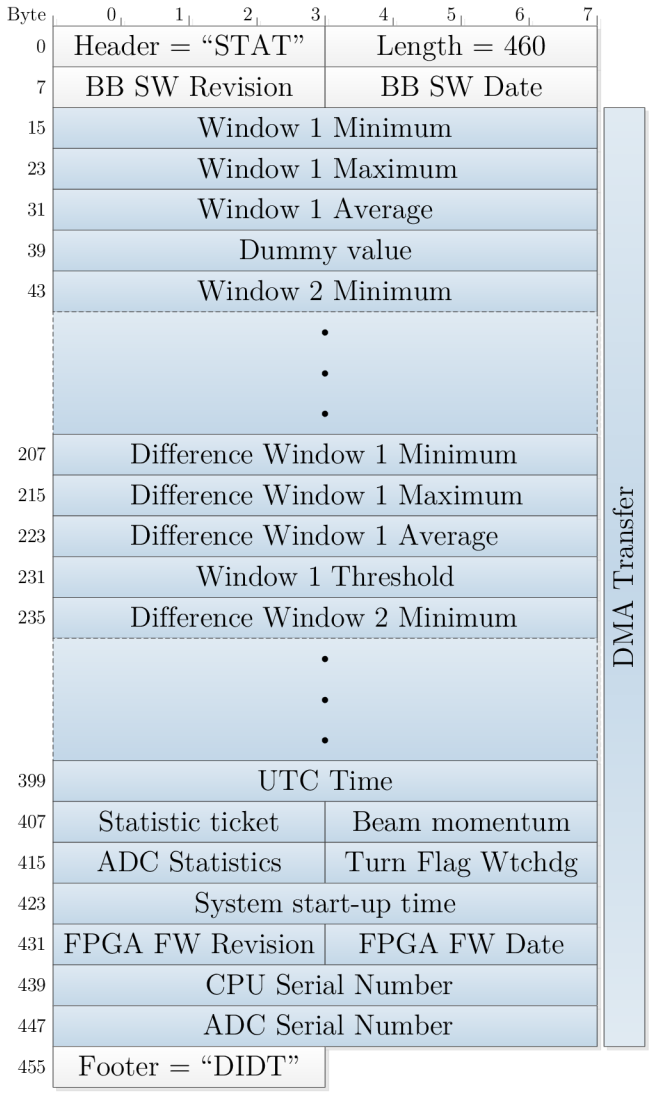



Fig. 4.7: Structure of the statistic message

The statistic message structure is shown in Fig. 4.7. The first two values of the message body are the software revision number and the compilation date. They are constant for all these messages. The BeagleBone fills them once with the message initialisation. They are followed by the DIDT Master statistic data. The first set of this data contains the statistic values of the moving average for all the windows. The other set contains statistically processed differences of all the windows. The end of the message body is filled with additional information:

- The time stamp and the statistic ticket have the same meaning like the corresponding values in the turn data messages.
- The beam momentum is the information decoded from the TTC or the GMT telegram.
- The ADC statistics represent the dynamic of the signal and actively used range of the ADC.
- The turn flag watchdog is the number of ADC clock ticks between two turn flags. It has to be constant.
- The system start-up time is the first time received via the telegram after the device starts.
- The CPU and ADC serial numbers are read out from the silicon serial number chips in the CPU and ADC subsystems.

### Beam Dump Message

The dump message body contains the time stamp of the beam dumping request and the identifier of the window which requested to dump the beam. The message structure is depicted in Fig. 4.8.

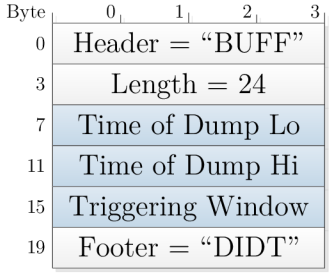



Fig. 4.8: Structure of the beam dump message

# 5 FBCCM MEASUREMENTS

## 5.1 Measurements of the RF Front End Characteristics

The RF Front End scattering parameters of all the fabricated ADC Subsystems had to be measured to verify their function and that they comply with the requirements. The measured scattering parameters were insertion gain $s_{21}$ and input return loss $s_{11}$. Their frequency characteristics were measured by a network analyser Agilent E5071C. This measurement is provided by the additional SMA connector at RF Front End output.

The comparison of the measured and simulated insertion gain is depicted in Fig. 5.1. The measured gain approximately follows the shape of the simulation output in the frequency band up to 100 MHz. The real RF Front End shows minor peak at the frequency around 200 MHz. This is probably caused by the resonance of the used RF coils. The measured frequency characteristics of the gain are sufficient, as the final filtration is performed digitally in the FPGA.

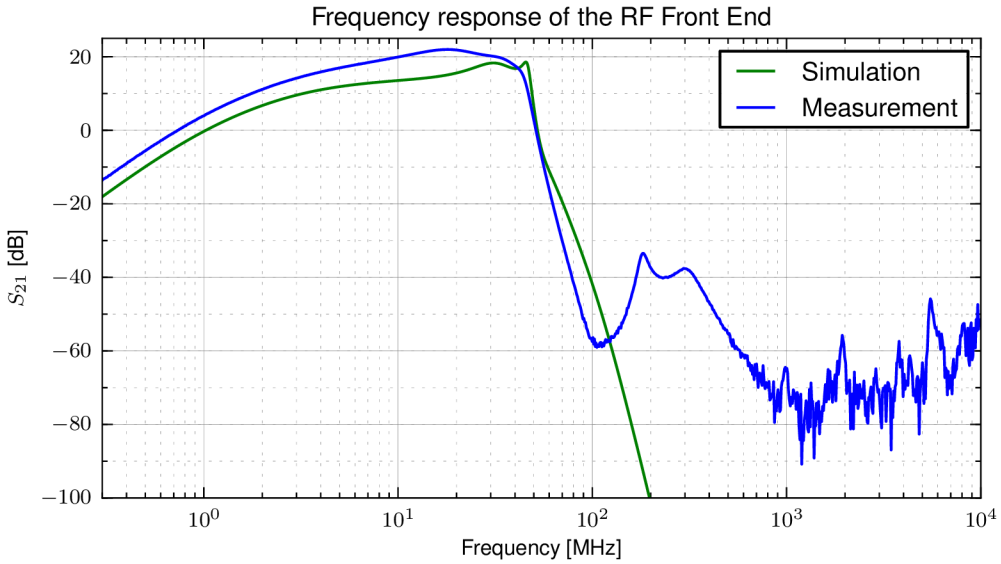



Fig. 5.1: Measurement of the RF Front End insertion gain

The measured input reflection is shown in Fig. 5.2. It is below -30 dB up to frequency of 100 MHz and does not show any major peak in the processed frequency band.

Tab. 5.1 shows evaluation of RF Front End parameters of all the fabricated modules. The measured parameters show that their tolerance is lower than the tolerance analyses stated (Sec. 2.1.2 RF Front End of ADC Subsystem).

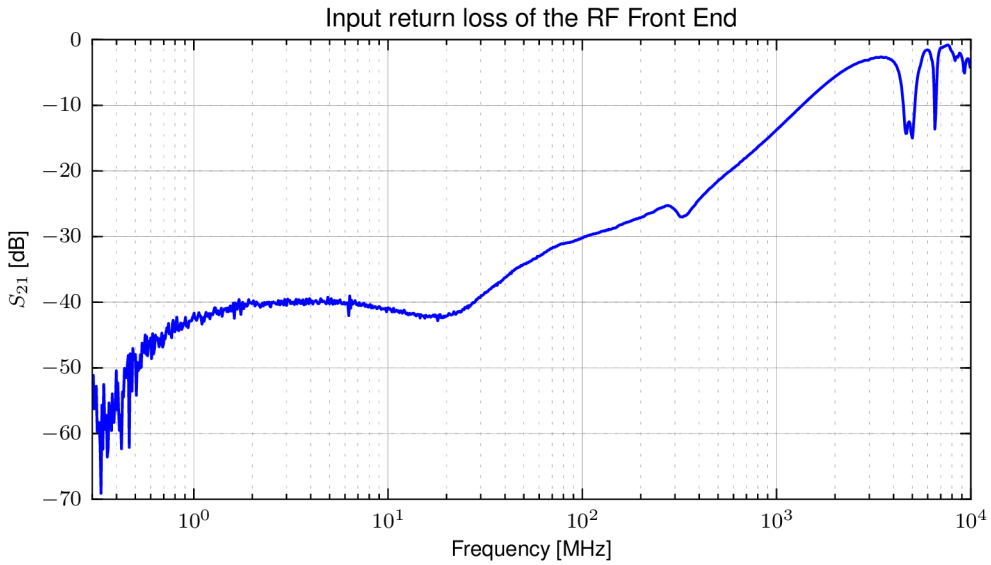



Fig. 5.2: Measurement of the RF Front End input return reflection

Tab. 5.1: Measured RF Front End frequency characteristics of all the ADC Subsystems

| **Module No.** | **Gain at 40 MHz** | **Maximum Gain** |
|---|---|---|
| 1 | 17.50 | 21.98 |
| 2 | 17.73 | 21.61 |
| 3 | 17.19 | 21.68 |
| 4 | 17.36 | 21.78 |
| 5 | 17.17 | 21.84 |
| 6 | 16.92 | 21.60 |
| 7 | 16.90 | 21.50 |
| 8 | 17.27 | 21.77 |
| 9 | 17.04 | 21.59 |
| 10 | 16.73 | 21.53 |
| 11 | 17.22 | 21.68 |
| 12 | 17.08 | 21.68 |

## 5.2 Laboratory Simulation of Losses

The FBCCM functionality had to be tested and verified in the laboratory before the FBCCM will be installed in the LHC. The laboratory testing should provide the FBCCM with signals and conditions which could occur during the machine operation.

The device under test has to be provided with the TTC optical signal and with an equivalent intensity signal. The intensity signal, which is normally provided by the FBCT, can be substituted by a 40 MHz sine-wave. This 40 MHz signal has to be synchronous with the TTC bunch clock which ensures the coherent signal sampling. The precise FBCT signal waveform is not required by the FBCCM, because it processes only the narrow frequency band. The 40 MHz signal amplitude has to be modulated to simulate changes of the beam intensity. The beam in the machine can circulate and slowly lose its intensity up to 24 hours before it is dumped. The amplitude modulation has to produce an equivalent slowly falling intensity shape. The input modulation signal has to be DC coupled to allow generation such slow changes of amplitude.

The environment used for the laboratory simulation of losses is shown in Fig. 5.3. The 40 MHz signal is generated by the sine-wave generator locked on to the TTC bunch clock. This signal amplitude is modulated by a variable voltage-controlled attenuator. The currently used attenuator is the ZX73-2500 from Mini-Circuits. The driving voltage for the attenuator is generated by the laboratory power supply Hameg HMP4040. This power supply is controlled over the Ethernet connection. The attenuation waveform is sent to the Hameg power supply by the Python application which is executed in the network computer.

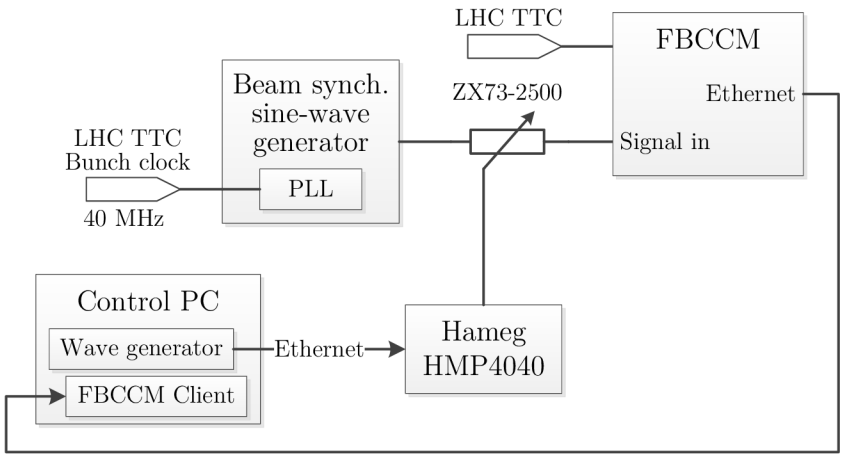



Fig. 5.3: Block diagram of the environment used for laboratory simulation of losses

The FBCCM has been tested by this method. The waveform generated by the Hameg power supply was linear ramp up and down. The transfer function of the variable attenuator is not linear. Therefore the 40 MHz signal amplitude measured

by the FBCCM is not linear as well. The generated 40 MHz signal amplitude will be linearised in the further tests.

Fig. 5.4 shows the simulation statistic data of window lengths 1, 4, 16. The statistic data of window lengths 64, 256, 1024 are depicted in Fig. 5.5. The graphs show the 10 second averages of the window differences and of the window moving average. The moving averages are labelled as 'AvgX' where X stands for the window number. The differences are similarly labelled as 'DiffX'. Each depicted difference is divided by the length of its window. This reduces spread of the difference traces over the Y-axis. The FBCCM observes the beam losses; therefore the difference is depicted with negation. The rising input signal amplitude induces a negative difference trace and the losses induce a positive trace.

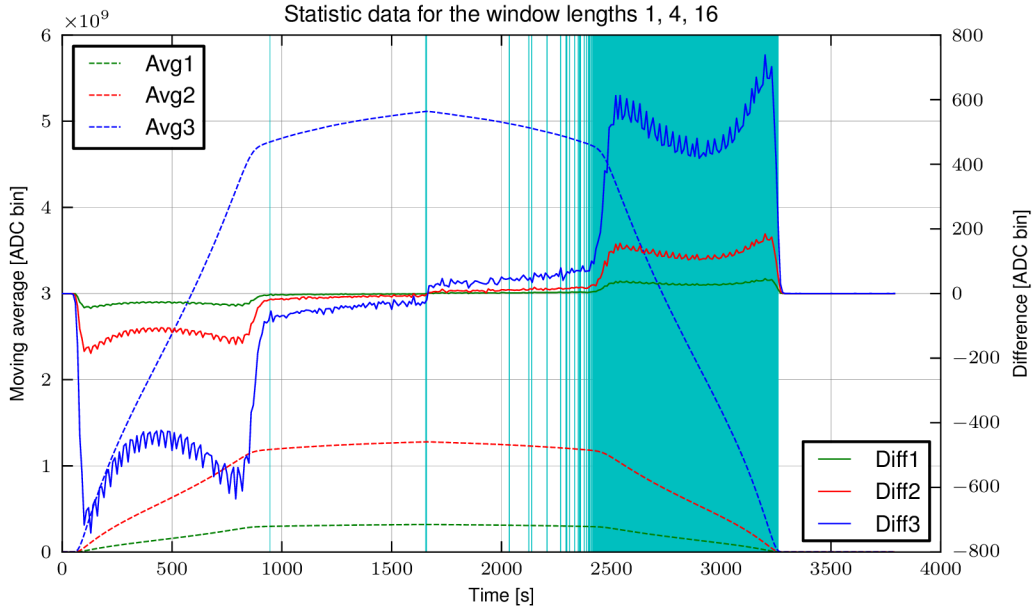



Fig. 5.4: Block diagram of the environment for loss simulation in the laboratory

All the corresponding signals have very similar shape for the different windows. The shape differences are caused by different length of integration. They are too small to be seen in these graphs. If the amplitude of the signal received by the FBCCM changes linearly in time, the waveform shapes are ideally same.

The difference waveforms have a saw shape in certain intervals. This is something unexpected and will be analysed in the future tests. The saw shape can be produced either naturally by the character of the generated FBCCM input signal or by a bug in the signal processing. If there was a bug in the system, it would likely be in the difference calculation, because only the difference values are visibly affected.

The vertical lines in figures represent the beam dumping requests which were triggered by the FBCCM. The emulated losses are very fast. The steep slope of the signal amplitude produces almost the continuous dumping requests. The beam

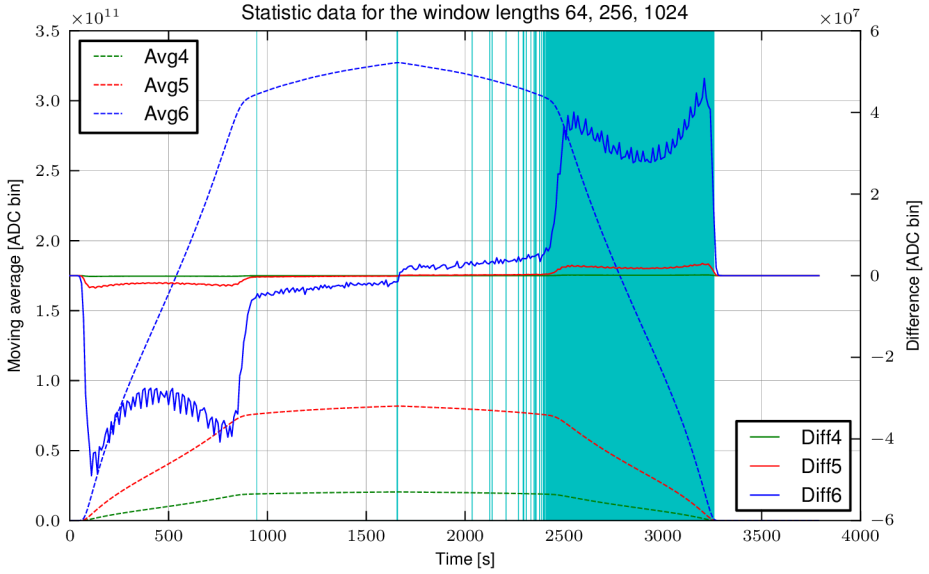



Fig. 5.5: Block diagram of the environment for loss simulation in the laboratory

dumping was also triggered when the signal amplitude was rising. This could be caused by thresholds set too stringently for the current environment, or they could be triggered due to a potential bug in the signal processing. This will have to be analysed as well in the future.

# 6 CONCLUSION

The ADC and CPU subsystems and the auxiliary power supply required by the ADC Subsystem were designed and implemented. Ten modules of all the subsystems of the final version were manufactured. All these modules were tested to verify their functionality. The RF Front End scattering parameters of the ADC Subsystem were measured by the network analyser. The average measurement outcomes are the 40 MHz gain of 17.2 dB and the amplifier bandwidth of $\approx$ 21 MHz. The RF Front End and the ADC were tested using a pure sine wave signal as the input signal to verify the analogue-to-digital conversion. The test outputs have been evaluated providing a satisfying average ENOB of 10.5 bits for the used 16 bit ADC. Besides these, many subsequent tests verified the complete functionality of all the hardware.

Simultaneously with the hardware development, the FPGA firmware and the BeagleBone software were developed. The FPGA firmware initialises the BeagleBone, processes the intensity measurement, and provides the BeagleBone with the processed data through the GPMC. The data is transferred into the BeagleBone using the DMA and sent to all clients through the TCP sockets. The DMA transfers are implemented to preserve the BeagleBone computing power for the future tasks.

The manufactured hardware modules were assembled together with the other parts to form six FBCCMs. All the constructed devices were loaded with the implemented FPGA firmware and the BeagleBone software. They were tested by a laboratory simulation of the beam losses to approve the desired FBCCM functionality of loss detection and triggering the beam dumping. The simulation provided the expected results, but for further tests the loss emulation will have to be enhanced. Simultaneously with further simulations, the BeagleBone performance tests are going to be performed. Their goal is to determine the maximum number of clients that the BeagleBone can reliably serve.

After the complete laboratory testing, four FBCCMs will be installed in July 2014 into the LHC Point 4. The connectivity of the installed FBCCMs in the LHC has to be verified, because of a different network infrastructure. Besides the connectivity test, the FBCCM will measure the machine noise (without the beam) to determine the noise background of the FBCCM in the LHC. The tests with the beam will be performed with the LHC start after the LS1. A dedicated LHC machine development time will be allocated to fully confirm the laboratory measurements by measurements in the real environment.

# REFERENCES


[1] H. Torres and G. Calderini, *Observation d'une nouvelle particule dans la recherche du boson de Higgs se desintegrant en deux photons dans l'experience ATLAS au LHC*. PhD thesis, Diderot U., Paris, 2013. Presented 2013.

[2] D. Boussard and col., "The lhc superconducting cavities," in *Proceedings of the 1999 Particle Accelerator Conference, New York, 1999*, pp. 946–948, 1999. Available: <http://cds.cern.ch/record/386691/files/mop120.pdf> [cit. 2014-05-03].

[3] L. R. Evans and P. Bryant, *The CERN Large Hadron Collider: Accelerator and Experiments*. Institute of Physics Publishing and SISSA, 2008.

[4] D. Belohrad and P. Kaspar, *Fast Beam Intensity Measurements for the LHC*. PhD thesis, Prague U., 2010. Presented on 05 Oct 2010.

[5] D. Belohrad and col., "Implementation of the Electronics Chain for the Bunch by Bunch Intensity Measurement Devices for the LHC," in *Proceedings of IPAC'10, Kyoto, Japan*, pp. 137–139, 2009. Available: <http://accelconf.web.cern.ch/accelconf/d09/papers/mopd43.pdf> [cit. 2013-11-27].

[6] B. Taylor, "Timing Distribution at the LHC." [Online], 2002. Available: <http://ttc.web.cern.ch/TTC/LECC02.pdf> [cit. 2013-11-27].

[7] J. Savioz, "The Beam Synchronous Timing Receiver InterfaceFor the Beam Observation." [Online], 2003. Available: <https://edms.cern.ch/file/406137/1.0/LHC-BOBR-ES-0001.pdf> [cit. 2013-11-27].

[8] J. Serrano and col., "Nanosecond level utc timing generation and stamping in cern's lhc," in *Proceedings of ICALPECS2003, Gyeongju, Korea*, pp. 119–121, 2003. Available: <http://accelconf.web.cern.ch/accelconf/ica03/PAPERS/MP533.PDF> [cit. 2014-04-27].

[9] B. Puccio and col., "The CERN Beam Interlock System: Principal and Operational Expirience," in *Proceedings of IPAC'10, Kyoto, Japan*, pp. 2866–2868, 2011. Available: <http://accelconf.web.cern.ch/accelconf/IPAC10/papers/wepeb073.pdf> [cit. 2013-11-25].

[10] CERN, "Beam dumping system." [Online], 2010. Available: <https://lhc-mp-review.web.cern.ch/lhc-mp-review/documents/LHC-DR-LBDS.pdf> [cit. 2013-11-25].

[11] B. Todd, "User Interface to the Beam Interlock System." [Online], 2006. Available: <http://ab-div-bdi-bl-blm.web.cern.ch/ab-div-bdi-bl-blm/Electronics/BLECS_Combiner/BLECS-Documents/BLECS-BeamPermit/CIBU-User-Manual-1v4.pdf> [cit. 2013-11-25].

[12] E. Gschwendtner and col., "Lhc beam loss monitors," in *Proceedings DIPAC 2001 – ESRF, Grenoble, France*, pp. 198–200, 2001. Available: <http://accelconf.web.cern.ch/accelconf/d01/papers/PM14.pdf> [cit. 2014-05-04].

[13] D. Belohrad and col., "The LHC Fast Beam Current Change Monitor," in *Proceedings of IBIC2013, Oxford, UK*, pp. 887–890, 2013. Available: <http://ibic2013.org/prepress/papers/wepf29.pdf> [cit. 2013-11-25].

[14] Texax Instruments, *ADS548xEVM – User's Guide*, Aug 2008. SLAU207.

[15] Analog Devices, *AN-935 Designing an ADC Transformer-Coupled Front End*, 2007. AN06982-0-9/07(0) Rev. 0.

[16] Texax Instruments, *ADS5484, ADS5485 – 16-bit, 170/200-MSPS Analog-to-Digital Converters*, Aug 2008. Rev. Oct 2009.

[17] Mini-Circuits, *ADT1-1WT – Surface mount RF transformer*, Aug 2012. Rev. D.

[18] Silicon Labs, *Si5326 – Any frequency precision clock multiplier/jitter atenuator*, Sep 2010. Rev. 1.0.

[19] Mini-Circuits, *ADT4-1WT – Surface mount RF transformer*, Nov 2007. Rev. C.

[20] Beagleboard.org, *BeagleBone Rev A6 System Reference Manual*, May 2012. Rev. 0.0.

[21] Linear Technology, *LT1764A Series*, July 2006. Rev. B.

[22] Texas Instruments, *PTN78060W, PTN78060H*, June 2009. SLTS229B.

[23] Acculam – Laminated Thermoset Plastic, *Acculam Epoxyglass G10/FR4 – Product Data Sheet*, Dec 2007.

[24] Texax Instruments, *AM335x ARM Cortex-A8 Microprocessors – Technical Reference Manual*, Nov 2012. Rev. G.

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ADC | Analogue to Digital Converter |
| BB | BeagleBone |
| BBE | BeagleBone Entity |
| BIS | Beam Interlock System |
| BLM | Beam Loss Monitor |
| BPF | Bandpass Filter |
| BST | Beam Synchronous Timing |
| CCC | CERN Control Centre |
| CERN | European Organization for Nuclear Research |
| CFI | Common Flash Interface |
| CIBU | Controls-Interlocks-Beam-User |
| CMOS | Complementary Metal-Oxide-Semiconductor |
| CPU | Central Processing Unit |
| DDR | Dual Data Rate |
| DIDT | Difference of intensity with respect to time |
| DMA | Direct Memory Access |
| DRAM | Dynamic Random Access Memory |
| EBL | Altera Ethernet Blaster |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| ESR | Equivalent Series Resistance |
| FBCCM | Fast Beam Current Change Monitor |
| FBCT | Fast Beam Current Transformer |
| FIR | Finite Impulse Response |
| FPGA | Field-Programmable Gate Array |
| FSM | Finite State Machine |
| GMT | General Machine Timing |
| GPMC | General Purpose Memory Controller |
| GPMC | General Purpose Memory Controller |
| HSMC | High Speed Mezzanine Card |
| I2C | Inter-Integrated Circuit |
| IC | Integrated Circuit |
| IP | Intellectual Property |
| IP | Internet Protocol |
| IRQ | Interrupt Request |
| JTAG | Joint Test Action Group |
| LDO | Low Drop Out |
| LED | Light-Emitting Diode |
| LHC | Large Hadron Collider |
| LOL | Loss of Lock |
| LOS | Loss of Signal |

| | |
|---|---|
| LPF | Lowpass Filter |
| LS1 | Long Shutdown 1 |
| LVDS | Low-Voltage Differential Signaling |
| MAC | Media Access Control |
| MMC | Multimedia Card |
| MOSFET | Metal-Oxide-Semiconductor Field-Effect Transistor |
| nADV | Negative Address Valid |
| nCS | Negative Chip Select |
| nOE | Negative Output Enable |
| PCB | Printed Circuit Board |
| PLL | Phase-Locked Loop |
| RAM | Random Access Memory |
| RF | Radio Frequency |
| ROM | Read Only Memory |
| RTL | Register-Transfer Level |
| SD | Secure Digital |
| SFP | Small Form-factor Pluggable |
| SMA | Sub-Miniature version A |
| SNR | Signal-to-Noise Ratio |
| SRAM | Static Random Access Memory |
| TCP | Transmission Control Protocol |
| TTC | Timing, Trigger and Control |
| TVS | Transient-Voltage-Suppression |
| UART | Universal Asynchronous Receiver Transmitter |
| XIP | Execute in Place |

# LIST OF FIGURES


1.1 CIBU input circuitry . . . . . 8
1.2 Diagram of the FBCCM system [13] . . . . . 10
1.3 The new FBCCM system concept . . . . . 13
2.1 Block diagram of the ADC Subsystem . . . . . 14
2.2 Block diagram of the analogue to digital conversion of the ADC Subsystem . . . . . 15
2.3 Schematic of the implemented RF Front End . . . . . 16
2.4 Simulation of the frequency response of the RF Front End . . . . . 16
2.5 Tolerance analysis of gain of ADC Subsystem RF Front End . . . . . 17
2.6 Tolerance analysis of the high-pass and the low-pass cut-off frequencies of the ADC front end . . . . . 18
2.7 Schematic of the equivalent circuit of the ADS5485 signal input [16] . 18
2.8 Schematic of the implemented matching circuit for the ADS5485 signal input . . . . . 19
2.9 The aggregated circuit used for the input impedance calculation of the matching network and the ADC signal input . . . . . 19
2.10 Block diagram of the sampling clock generation . . . . . 21
2.11 Schematic diagram of ADC clock filter . . . . . 22
2.12 Simulated frequency response of the Clock Low-Pass Filter . . . . . 22
2.13 Simulation of the low cut-off frequency tolerance of the Clock Low-Pass Filter . . . . . 23
2.14 Simulation of the gain tolerance of the Clock Low-Pass Filter . . . . . 23
2.15 Equivalent ADS5485 clock input circuit [16] . . . . . 25
2.16 Schematic of the equivalent simplified circuit of the ADS5485 clock input pin . . . . . 26
2.17 Balun transformer and matching load of the ADC clock signal . . . . 27
2.18 Block diagram of the TTC reciever . . . . . 28
2.19 Bidirectional voltage level translator for $I^2C$ . . . . . 29
2.20 Picture of the front panel indicators and controls . . . . . 29
2.21 Block diagram of the CPU Subsystem . . . . . 30
2.22 BeagleBone microprocessor module . . . . . 31
2.23 Implemented output circuit for the client CIBU interface . . . . . 32
2.24 Flexible connection between the CPU board and the back panel connector for the CIBU . . . . . 33
2.25 Circuit for the clock input of the CPU board . . . . . 34
2.26 Schematic of the clock output amplifier of the CPU Subsystem . . . . 34
2.27 FBCCM power system . . . . . 35
2.28 Block diagram of the ADC Auxiliary Power Supply . . . . . 36
2.29 Block diagram of power distribution of the ADC Subsystem . . . . . 37
2.30 Schematic of the optocoupler shutdown circuit . . . . . 38

2.31 Block diagram of the power distribution in the CPU Subsystem . . . 39
2.32 PCB stack for the ADC Subsystem . . . . . 40
3.1 Block diagram of the FPGA firmware . . . . . 43
3.2 Block diagram of the DIDT Master . . . . . 44
3.3 FIFO test of the non-zero value propagation . . . . . 46
3.4 FIFO test of the ramp propagation . . . . . 47
3.5 Implementation of the moving average into the FPGA . . . . . 47
3.6 Moving average test of the first maximum and minimum after reset . 49
3.7 Moving average test of the blanking interval . . . . . 50
3.8 Moving average test of the last sample registration . . . . . 50
3.9 Moving average test of the window length . . . . . 51
3.10 Moving average test of the ramp up and down . . . . . 52
3.11 Moving average test of the long ramp up . . . . . 53
3.12 Moving average test of the long ramp down . . . . . 53
3.13 Implementation of the turn flag synchronisation circuit . . . . . 54
3.14 Example waveforms of the implemented synchronisation circuit . . . . 54
3.15 Block diagram of the BeagleBone entity . . . . . 57
3.16 Asynchronous single read access with wait pin monitoring [24] . . . . 58
3.17 State diagram of the GPMC Slave . . . . . 59
3.18 Mealy FSM state diagram convention . . . . . 59
4.1 Bootloader flow chart . . . . . 62
4.2 Block diagram of the main application . . . . . 63
4.3 DMA descriptors for the ping-pong buffering . . . . . 64
4.4 Flow chart of the DMA interrupt processing . . . . . 64
4.5 Flow chart of the application main loop . . . . . 65
4.6 Structure of the turn data message . . . . . 67
4.7 Structure of the statistic message . . . . . 68
4.8 Structure of the beam dump message . . . . . 69
5.1 Measurement of the RF Front End insertion gain . . . . . 70
5.2 Measurement of the RF Front End input return reflection . . . . . 71
5.3 Block diagram of the environment used for laboratory simulation of losses . . . . . 72
5.4 Block diagram of the environment for loss simulation in the laboratory 73
5.5 Block diagram of the environment for loss simulation in the laboratory 74
A.1 Schematic diagram of the ADC Subsystem . . . . . 84
A.2 Schematic diagram of the CPU Subsystem . . . . . 85
A.3 Schematic diagram of the ADC Auxiliary Power Supply . . . . . 86
A.4 Perspective view of the ADC Subsystem module . . . . . 87
A.5 Top view of the ADC Subsystem module . . . . . 88
A.6 Bottom view of the ADC Subsystem module . . . . . 88
A.7 Perspective view of the CPU Subsystem module . . . . . 89
A.8 Top view of the CPU Subsystem module . . . . . 90

A.9 Bottom view of the CPU Subsystem module . . . . . 90
A.10 Top view of the ADC Auxiliary Power Supply . . . . . 91
A.11 Bottom view of the ADC Auxiliary Power Supply . . . . . 92
A.12 Mechanical model of the FBCCM device . . . . . 93
A.13 Detail view of the CPU Subsystem model . . . . . 94
A.14 Perspective view of the FBCCM device without top panel . . . . . 95
A.15 Top view of the FBCCM device without top panel . . . . . 96
A.16 Photograph of the FBCCM front panel . . . . . 97
A.17 Photograph of the FBCCM back panel . . . . . 97

# LIST OF TABLES

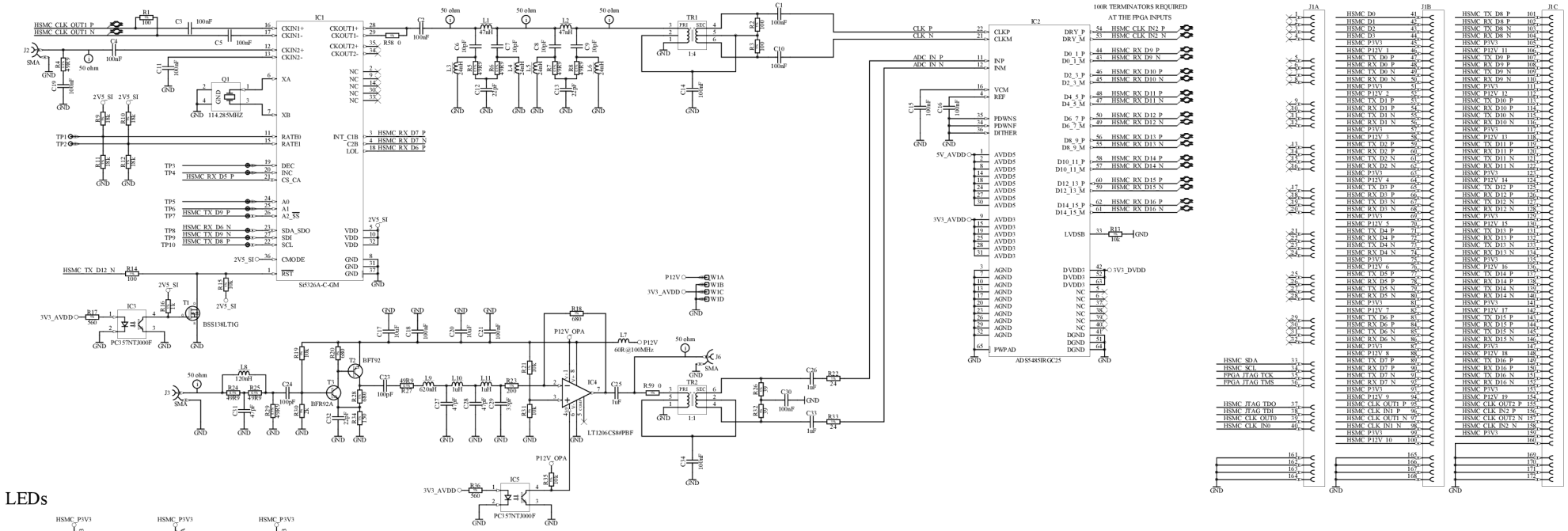
5.1 Measured RF Front End frequency characteristics of all the ADC Subsystems . . . . . 71

# APPENDIXES

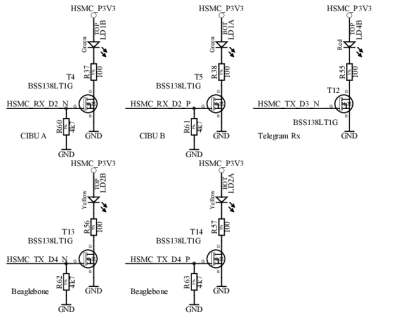
## A.1 Schematic Diagrams of FBCCM Subsystems

Subsequent pages contain schematic diagrams of the FBCCM subsystems in the following order:

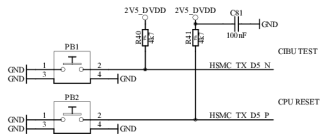
- Fig. A.1: Schematic diagram of the ADC Subsystem
- Fig. A.2: Schematic diagram of the CPU Subsystem
- Fig. A.3: Schematic diagram of the ADC Auxiliary Power Supply



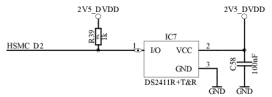
## LEDs



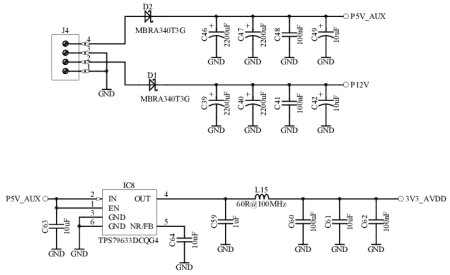
## TEST BUTTONS



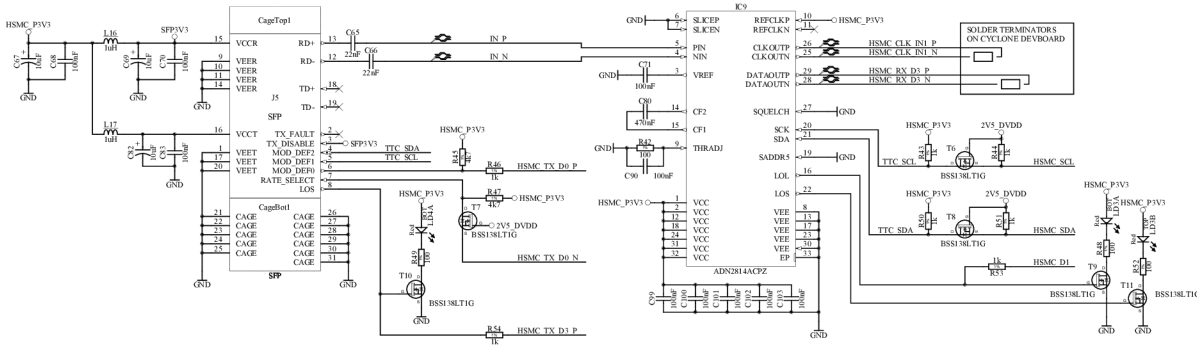
## SILICON SERIAL NUMBER

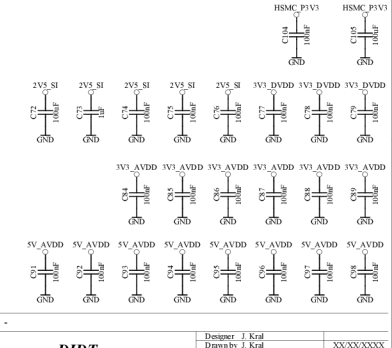


## EXTERNAL POWER SUPPLY

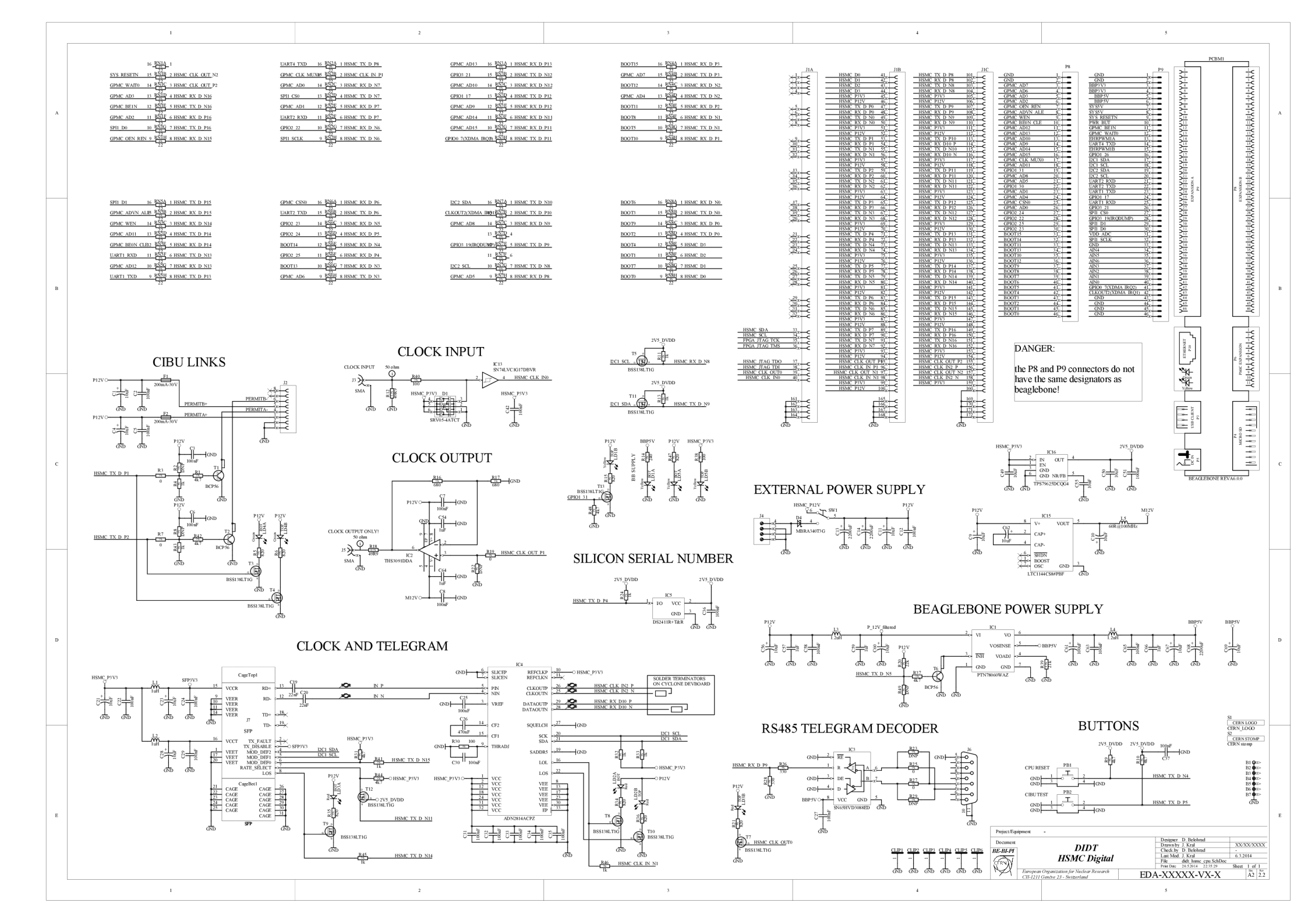


## CLOCK AND TELEGRAM

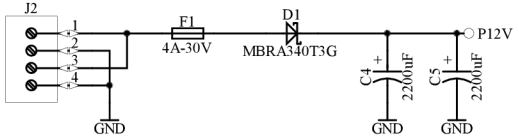




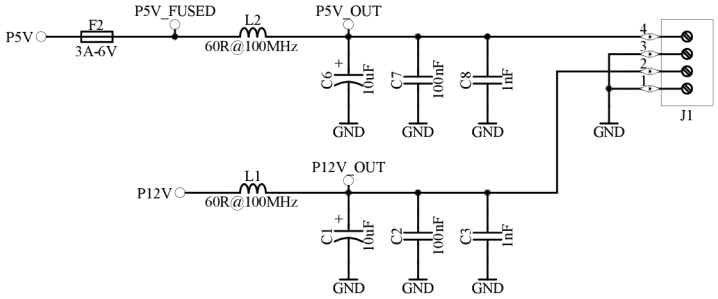
| | |
|---|---|
| Project/Equipment | - |
| Document | |
| **DIDT** | |
| **HSMC ADC module** | |
| Designer J. Kral | |
| Drawn by J. Kral | XX/XX/XXXX |
| Check. by D. Belohrad | - |
| Last Mod. J. Kral | 20.5.2014 |
| File dldt_hsmc_adc.SchDoc | |
| Print Date 20.5.2014 22:40:09 | Sheet 1 of 1 |
| | Size A2 Rev 1.2 |

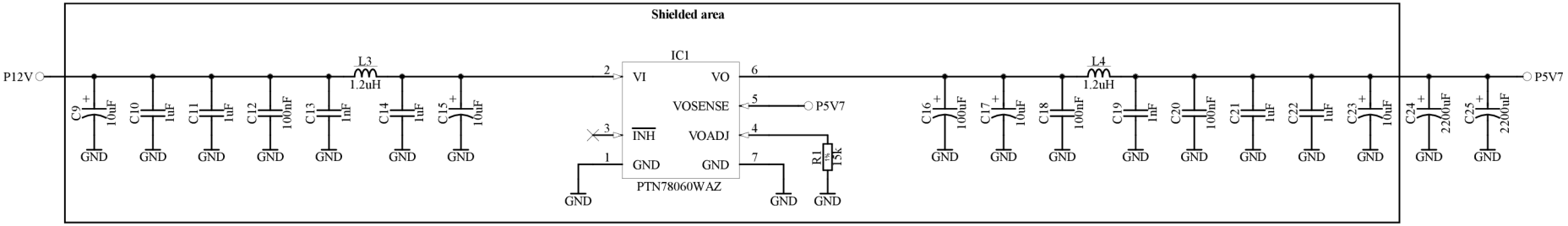

SYS_RESETN
GPMC_WAIT0
GPMC_AD3
GPMC_BE1N
GPMC_AD2
SPI1_D0
GPMC_OEN_REN
UART4_TXD
GPMC_CLK_MUX0
GPMC_AD0
SPI1_CS0
GPMC_AD1
UART2_RXD
GPIO2_22
SPI1_SCLK
GPMC_AD13
GPIO3_21
GPMC_AD10
GPIO1_17
GPMC_AD9
GPMC_AD14
GPMC_AD15
GPIO0_7(XDMA_IRQ0)
BOOT15
GPMC_AD7
BOOT12
GPMC_AD4
BOOT11
BOOT8
BOOT5
BOOT10
SPI1_D1
GPMC_ADVN_ALE
GPMC_WEN
GPMC_AD11
GPMC_BE0N_CLE
UART1_RXD
GPMC_AD12
UART1_TXD
GPMC_CSN0
UART2_TXD
GPIO2_23
GPIO2_24
BOOT14
GPIO2_25
BOOT13
GPMC_AD6
I2C2_SDA
CLKOUT2(XDMA_IRQ1)
GPMC_AD8
GPIO1_19(IRQDUMP)
I2C2_SCL
GPMC_AD5
BOOT6
BOOT3
BOOT9
BOOT2
BOOT4
BOOT1
BOOT7
BOOT0
CIBU LINKS
CLOCK INPUT
CLOCK OUTPUT
CLOCK AND TELEGRAM
SILICON SERIAL NUMBER
EXTERNAL POWER SUPPLY
BEAGLEBONE POWER SUPPLY
RS485 TELEGRAM DECODER
BUTTONS
DANGER:
the P8 and P9 connectors do not have the same designators as beaglebone!
BEAGLEBONE REVA6.0.0
SOLDER TERMINATORS ON CYCLONE DEVBOARD
DIDT
HSMC Digital
BE-BI-PI
European Organization for Nuclear Research
CH-1211 Genève 23 - Switzerland
EDA-XXXXX-VX-X

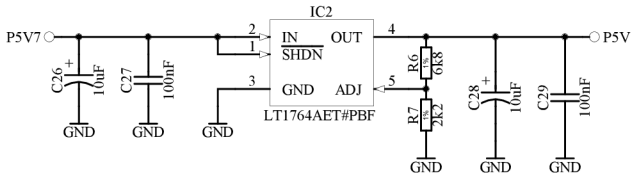
# POWER INPUT (max. 15 V)



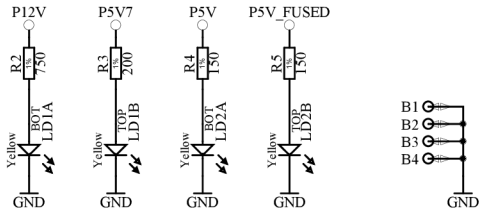

# POWER OUTPUT (Vin@max4A, 5V@3A)

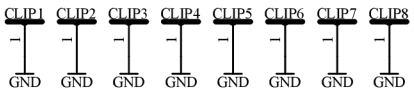



S1
CERN STOMP
CERN stomp
S2
CERN LOGO
CERN_LOGO




| | | | |
|---|---|---|---|
| Project/Equipment | - | | |
| Document | ***DIDT*** ***Power module*** | Designer J. Kral | |
| BE/BI | | Drawn by J. Kral | XX/XX/XXXX |
| | | Check.by D. Belohrad | - |
| | | Last Mod. J. Kral | 6.3.2014 |
| | | File didt_power_distr.SchDoc | |
| | | Print Date 20.5.2014 22:23:46 | Sheet 1 of 1 |
| * * | | | Size A4 / Rev 1.1 |

## A.2 Photographs of FBCCM Subsystems

### A.2.1 Photographs of the ADC Subsystem


Fig. A.4: Perspective view of the ADC Subsystem module

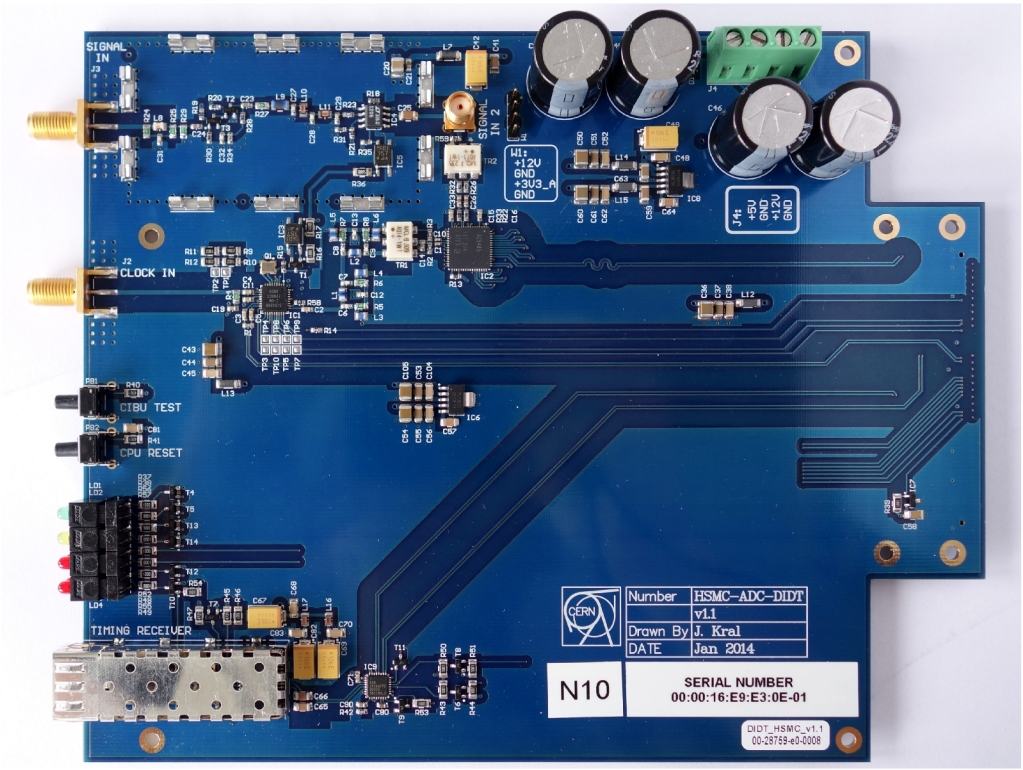



Fig. A.5: Top view of the ADC Subsystem module

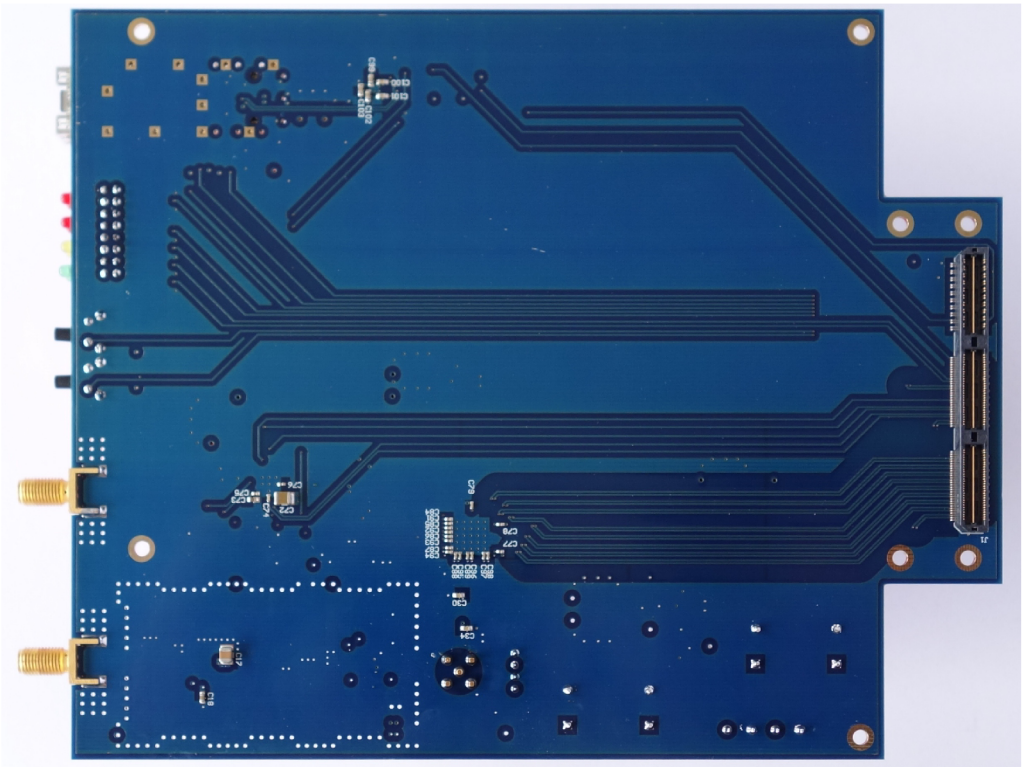


Fig. A.6: Bottom view of the ADC Subsystem module

### A.2.2 Photographs of the CPU Subsystem

The following photographs show the assembled CPU Subsystem module. The SMA connectors are missing, because they are fixed in the assembled FBCCM device box.


Fig. A.7: Perspective view of the CPU Subsystem module

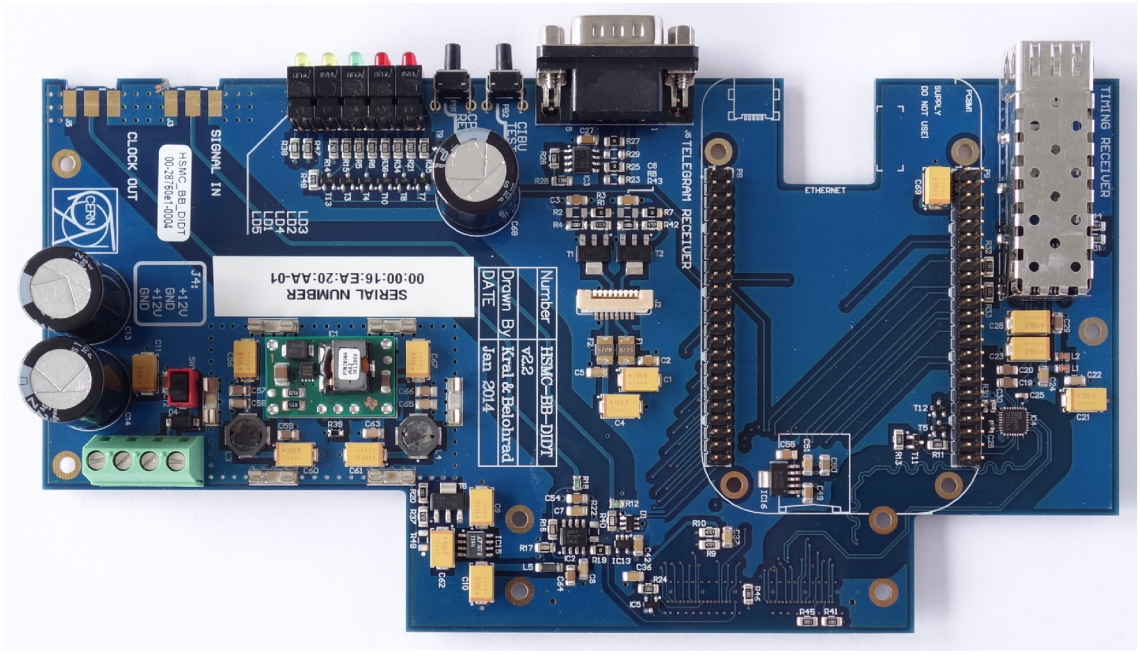



Fig. A.8: Top view of the CPU Subsystem module

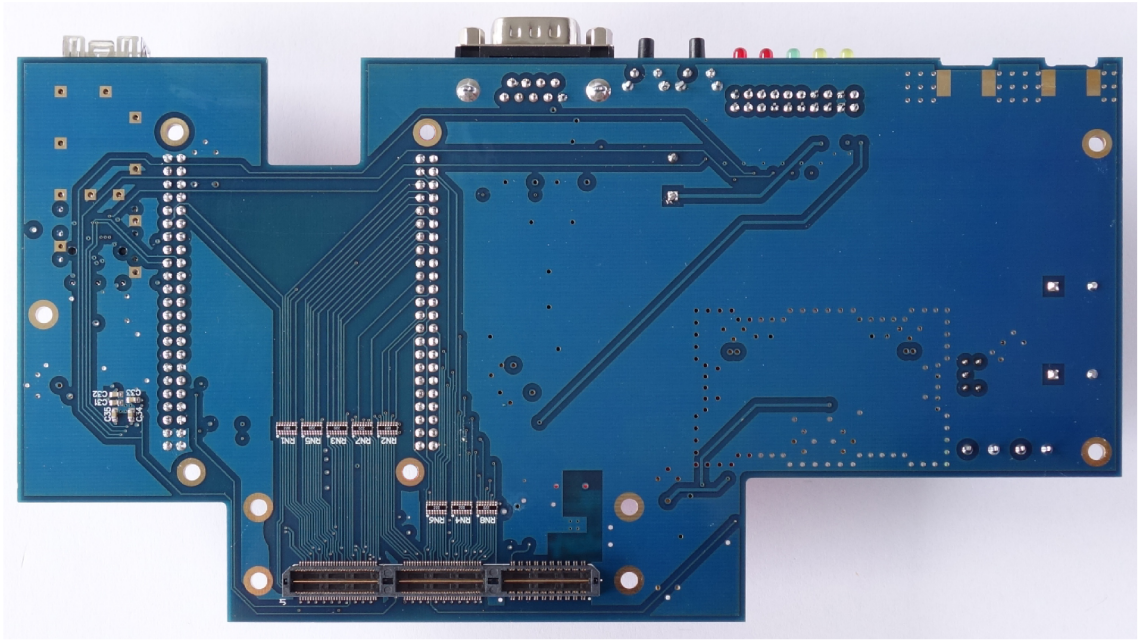


Fig. A.9: Bottom view of the CPU Subsystem module

### A.2.3 Photographs of the ADC Auxiliary Power Supply

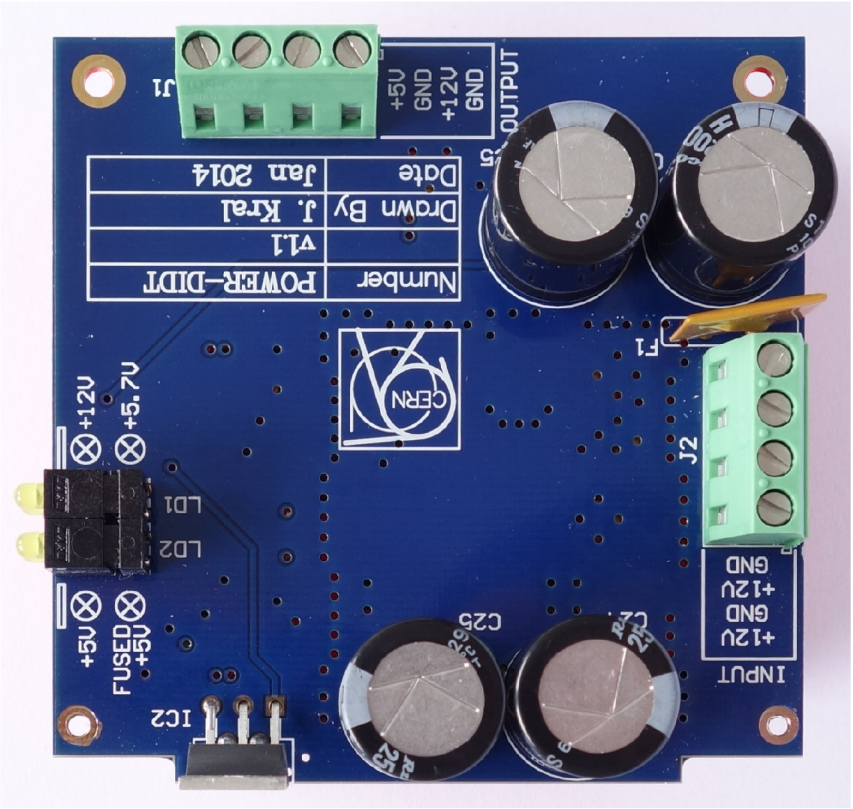



Fig. A.10: Top view of the ADC Auxiliary Power Supply

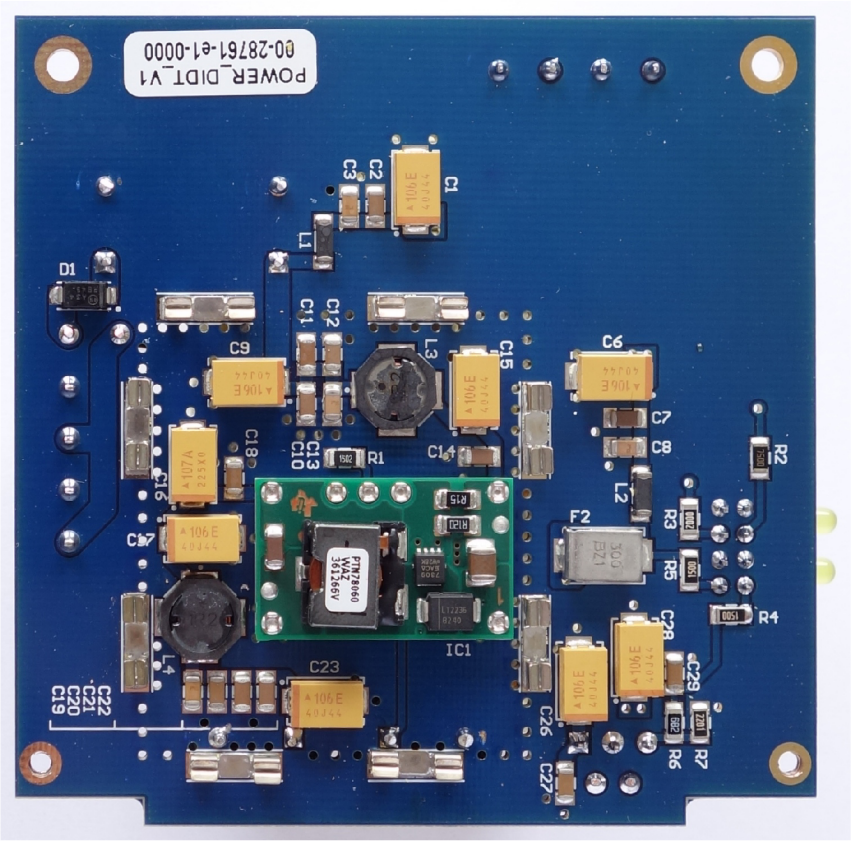



Fig. A.11: Bottom view of the ADC Auxiliary Power Supply

## A.3 Pictures of the FBCCM Mechanical Model

The following pictures depict the FBCCM CAD model designed in Autodesk Inventor.

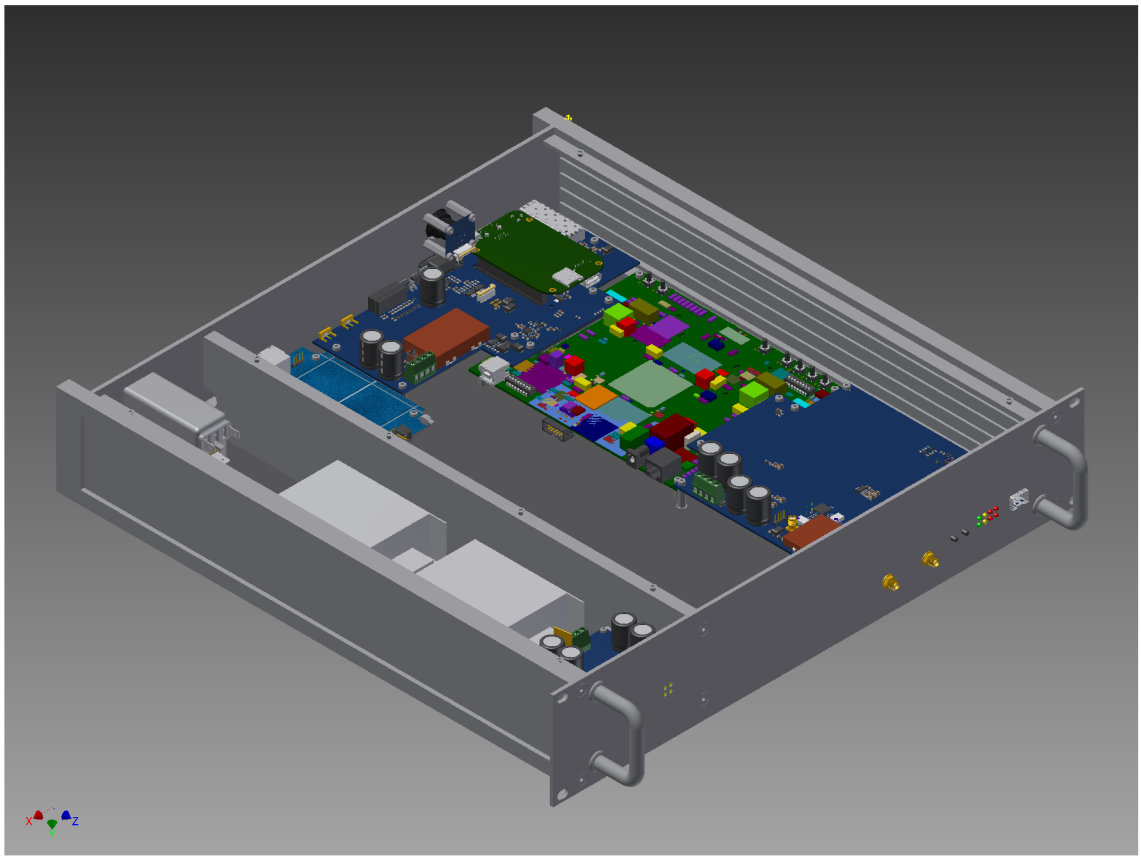


Fig. A.12: Mechanical model of the FBCCM device

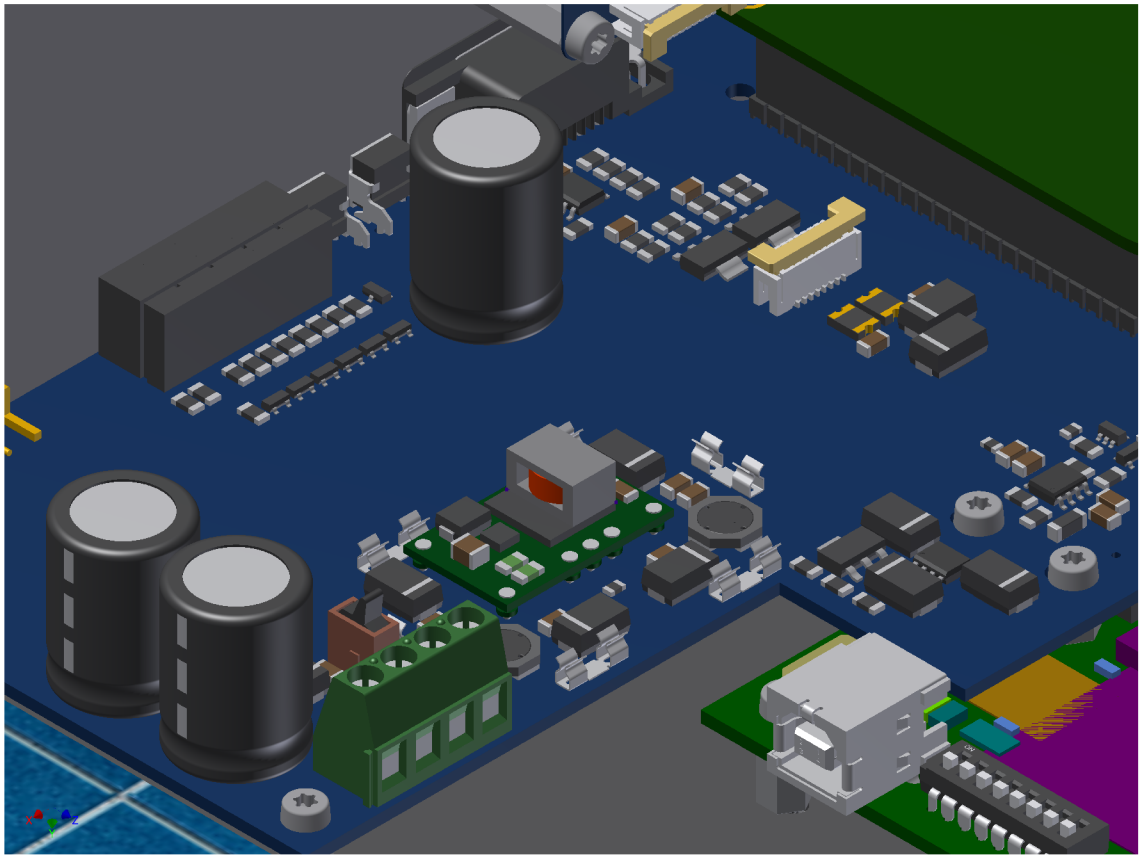


Fig. A.13: Detail view of the CPU Subsystem model

## A.4 Photographs of the FBCCM Device


Fig. A.14: Perspective view of the FBCCM device without top panel


Fig. A.15: Top view of the FBCCM device without top panel

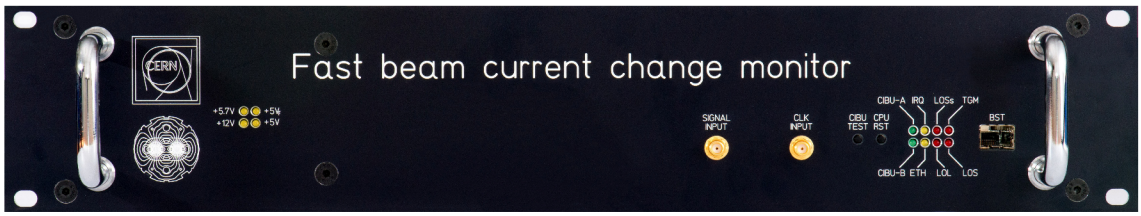



Fig. A.16: Photograph of the FBCCM front panel


Fig. A.17: Photograph of the FBCCM back panel

## A.5 VHDL Codes

### A.5.1 FIFO Buffer Entity

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- the input sample is propagated throught FIFO after
-- FIFO_DEPTH enable impulses

-- Example for FIFO_DEPTH = 4
--           _   _   _   _   _   _   _   _   _   _   _   _   _
-- ClkxC  _/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/
--            ___     ___     ___     ___     ___     ___     ___
-- DxE    _/   \___/   \___/   \___/   \___/   \___/   \___/   \
--        _____ _______ ___________________________________________
-- DxD    _0___X___1___X___0_______0_______0_______0_______0____
--                                    _______ ________
-- QxD    _0_______0_______0_______0_______0___X___1___X___0____

entity fifo is
  generic (
    FIFO_DEPTH : natural := 4;          -- FIFO depth
    DATA_WIDTH : natural := 8           -- FIFO data width
    );
  port (
    ClkxC : in  std_logic;              -- input clock signal
    DxE   : in  std_logic;              -- input enable signal
    DxD   : in  std_logic_vector(DATA_WIDTH-1 downto 0); -- data in
    QxD   : out std_logic_vector(DATA_WIDTH-1 downto 0) -- data out
    );
end entity fifo;


architecture fifo_v1 of fifo is
  -- Build a 2-D array type for the FIFO
  subtype word_t is std_logic_vector(DATA_WIDTH-1 downto 0);
  type fifo_t is array(FIFO_DEPTH-1 downto 0) of word_t;

  -- Declare the RAM signal.
  signal FifoxD : fifo_t := (others => (others => '0'));
```

```
begin  -- architecture fifo_v1

  -- purpose: shifts fifo memory each time enable signal is set
  -- type   : sequential
  -- inputs : ClkxC, ResetxRNA, DxE, DxD, FifoxD
  -- outputs: FifoxD
  fifo_shift : process (ClkxC) is
  begin  -- process fifo_shift
    if rising_edge(ClkxC) then          -- rising clock edge
      if DxE = '1' then
        -- new incoming data comes to the lowest position
        -- the whole fifo is left shifted
        if FifoxD'high > FifoxD'low then
          FifoxD(FifoxD'high downto FifoxD'low+1) <=
      FifoxD(FifoxD'high-1 downto FifoxD'low);
        end if;

        FifoxD(FifoxD'low) <= DxD;
      end if;
    end if;
  end process fifo_shift;

  -- assign the highest position to output data word
  QxD <= FifoxD(FifoxD'high);

end architecture fifo_v1;
```

### A.5.2 Flag Synchronisation Entity

```
library ieee;
use ieee.std_logic_1164.all;

entity sync_impulse is

  port (
    ClkxC0     : in  std_logic;  -- clock of source domain
    ResetxRNA0 : in  std_logic;  -- reset in source domain
    ClkxC1     : in  std_logic;  -- clock of destination domain
    ResetxRNA1 : in  std_logic;  -- reset in destination domain
    DxE0       : in  std_logic;  -- input impulse to be synchronise
    QxE1       : out std_logic   -- output synchronized impulse
    );

end entity sync_impulse;

architecture sync_impulse_v1 of sync_impulse is
  -- toggle signal changes its value when input impulse is detected
  signal TogglexS0  : std_logic;
  -- toggle signal after first synchronizing flip flog
  signal LatchedxS1 : std_logic;
  -- toggle signal synchronized to destination clock domain
  signal TogglexS1  : std_logic;
  -- delayd toggle signal for the edge detection
  signal DelayedxS1 : std_logic;
```

```
begin  -- architecture sync_impulse_v1

  -- purpose: toggle signal for synchronization
  -- type   : sequential
  -- inputs : ClkxC0, ResetxRNA0, DxE0
  -- outputs: TogglexS0
  INLATCH : process (ClkxC0, ResetxRNA0) is
  begin  -- process INLATCH
    if ResetxRNA0 = '0' then      -- asynchronous reset (active low)
      TogglexS0 <= '0';
    elsif rising_edge(ClkxC0) then   -- rising clock edge
      if DxE0 = '1' then
        -- change the value of toggle signal
        -- when input impulse is detected
        TogglexS0 <= not TogglexS0;
      end if;
    end if;
  end process INLATCH;

  -- purpose: synchronises Toggle signal to destination domain
  -- type   : sequential
  -- inputs : ClkxC1, ResetxRNA1, TogglexS0
  -- outputs: TogglexS1
  FlipFlopSync : process (ClkxC1, ResetxRNA1) is
  begin  -- process FlipFlopSync
    if ResetxRNA1 = '0' then      -- asynchronous reset (active low)
      LatchedxS1 <= '0';
      TogglexS1  <= '0';
    elsif rising_edge(ClkxC1) then      -- rising clock edge
      TogglexS1  <= LatchedxS1;
      LatchedxS1 <= TogglexS0;
    end if;
  end process FlipFlopSync;

  -- purpose: detects edges of toggle signal
  -- type   : sequential
  -- inputs : ClkxC, ResetxRNA, TogglexS1
  -- outputs: QxE1
  EdgeDetector : process (ClkxC1, ResetxRNA1) is
  begin  -- process EdgeDetector
    if ResetxRNA1 = '0' then      -- asynchronous reset (active low)
      DelayedxS1 <= '0';
      QxE1       <= '0';
    elsif rising_edge(ClkxC1) then      -- rising clock edge
      QxE1       <= DelayedxS1 xor TogglexS1;
      DelayedxS1 <= TogglexS1;
    end if;
  end process EdgeDetector;

end architecture sync_impulse_v1;
```