Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea Magistrale in Fisica

A PCIe-based readout and control board to interface with new-generation detectors for the LHC upgrade

Relatore:
Prof. Alessandro Gabrielli

Presentata da:
Claudio Preti

Correlatore:
Dott. Davide Falchieri

Anno Accademico 2015/2016

Abstract

Questa tesi si riferisce principalmente al lavoro di design, sviluppo, produzione e validazione di una nuova scheda PCIe, chiamata Pixel-ROD (Pixel Read Out Driver), come naturale prosecuzione della precedente serie di schede di readout, oggi montate nel Pixel Detector di ATLAS. In modo particolare, questa scheda è stata pensata come evoluzione per l'elettronica off-detector presente ad ATLAS, la quale è principalmente composta da schede VME, conosciute come Back Of Crate (BOC) e Read Out Driver (ROD). Inoltre, tutte le schede ROD sono state commissionate e disegnate dal Laboratorio di Progettazione Elettronica dell'INFN e del DIFA a Bologna.

Il progetto della scheda Pixel-ROD è cominciato due anni fa, poichè il trend generale per l'evoluzione dell'elettronica off-detector di LHC è quello di abbandonare la più vecchia interfaccia VME, per passare a quelle più nuove e veloci (come il PCIe). Inoltre, poichè i rivelatori di ATLAS e CMS saranno accomunati dallo stesso chip di readout che interfaccerà i futuri Pixel Detector, la Pixel-ROD potrebbe essere usata non solo per l'evoluzione di ATLAS ma anche per altri esperimenti.

La caratteristica principale della Pixel-ROD è la possibilità di utilizzo sia come scheda di readout singola, sia in una catena reale di acquisizione dati, che si interfaccia con dispositivi di terze parti.

Il lavoro che ho svolto in questa tesi si concentra principalmente sul design, lo sviluppo e l'ottimizzazione della scheda prima della sua fabbricazione. Dopo questa fase, utilizzando i prototipi prodotti, mi sono concentrato sul lavoro di test e validazione dei singoli componenti e delle singole interfacce montate sulla scheda. Questa fase non è ancora terminata e richiede molto tempo per essere svolta, a causa della complessità dell'elettronica che è presente sulla Pixel-ROD.

Contents

Introduction

This thesis mainly refers to the design, the development, the production and validation of a new PCIe board, named Pixel-ROD (Pixel Read Out Driver), as a natural follow-up of the previous series of readout boards, implemented into ATLAS Pixel detector. Particularly, this board was designed as an upgrade for the current off-detector electronics present at ATLAS, which is mainly made up of VME boards, known as Back of Crate (BOC) and Read Out Driver (ROD). In addition, all ROD boards have been designed and commissioned by the Electronic Design Laboratory of INFN and DIFA in Bologna. The project of Pixel-ROD board started a couple of years ago, since the general trend on the update for the off-detector LHC phase 2 electronics is to leave the older VME interface for newer and faster buses (such as PCIe). Moreover, as the ATLAS and CMS experiments will share the same readout chip that will interface the future Pixel Detectors, the Pixel-ROD board could be used not only for the ATLAS upgrade, but also for other experiments.

The main feature of the Pixel ROD board is that it can be used both as a standalone readout electronics or in real data acquisition chains by interfacing with third party devices.

This thesis is intended to provide a brief overview of the environment in which Pixel-ROD board was conceived. In particular, after this Introduction, the Chapter One summarizes the ATLAS experiment, focusing on the detectors point of view. Chapter Two shows, step by step, how the Pixel-ROD has been designed and intended. Chapter Three describes the tests that have been carried out so far. Finally, the Conclusions depict the current situation of tests, the obtained results and the future analysis that will be performed in order to match a real application on a specific experiment.

At this time, 2 Pixel-ROD prototype boards have been produced and are now being validated. The process of validation aims to verify board's full functionality by config-

uring, debugging and testing each device present on the board.

In particular, my work in this thesis has been mainly focused on the design, the development and the optimization of the board before its fabrication. After that, on the prototype boards, I focused on the tests and validations of the individual devices and interfaces mounted on the board. This phase is still on-going and it is particularly time consuming, due to the complexity of the electronics involved.

Chapter 1

LHC

LHC (Large Hadron Collider) is the largest particle accelerator ever built, placed in the tunnel which housed LEP (Large Electron-Positron collider) in Geneva, near the French-Swiss border. It is managed by the European Organization for Nuclear Research, also known as CERN (Conseil Européen pour la Recherche Nucleaire), which is a collaboration among 22 member states plus other "observers" non-member states from around the world.

LHC is made up of a ring 27 km long [1], which is placed at a medium depth of about 100 m. There are four main points (see Figure 1.1) of interaction where protons are forced to collide. At these points, huge detectors (known as ATLAS, ALICE, CMS and LHCb) are set up to record every detail of the particle collisions, providing a tremendous amount of data to analyse.

Protons are not straightly inserted into the beam pipe of the main ring, but they undergo a sequence of accelerators thanks to which they reach the desired energy. Firstly, thanks to a linear accelerator named LINAC 2, protons reach the energy of 50 MeV; after that they are fed to the Proton Synchrotron Booster (PSB) through which they accelerate until 1.4 GeV. A second synchrotron, named Proton Synchrotron (PS) pushes the beam to an energy of 25 Gev; finally, protons are brought to 450 GeV and are ready to be inserted in the beam pipes of LHC. Here, they are accelerated to 6.5 TeV thanks to radio frequency cavities working at 400 MHz.

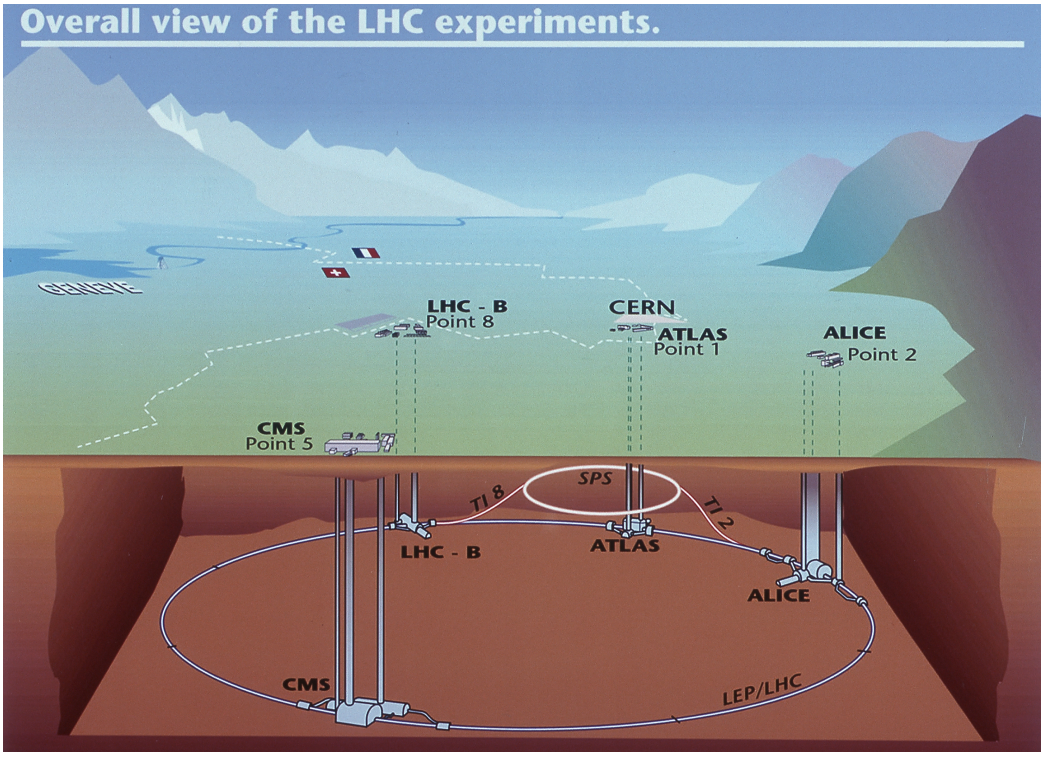


Figure 1.1: LHC overview.

1.1 Accelerator's parameters

The nominal maximum collision energy for protons in LHC is 14 TeV, however the accelerator is now working with a collision energy of 13 Tev, that is 6.5 TeV per proton beam. The reason lying behind this fact is that, with this convenient expedient, powering up the magnets would take less time, since a lower magnetic field would be necessary. In this way the delivery of particle for physical experiments is optimized, thereby speeding up the route to potential new discoveries. At this energy, protons move with a speed very close to the speed of light in vacuum. In LHC, under nominal operating conditions, each beam has 2808 bunches of protons, containing about 10^{11} particles each, allowing many proton's collisions at every bunch crossing. The beam is held in the accelerator ring by 1232 superconducting dipole magnets, that create a maximum magnet field of 8.3 T, and focused by 392 quadrupole magnets. The Large Hadron Collider is built to have a peak of instantaneous luminosity of $L = 10^{34} cm^{-2} s^{-1}$.

1.2 Main experiments at LHC

As already stated, there are four main points of interaction along LHC's ring, each one hosting a detector in which particles (protons or, for about one month per year, ions), are forced to collide.

- **ATLAS, A Toroidal Lhc ApparaTus:** it is one of the two general purpose detectors at LHC. This detector investigates a wide range of physical problems, spacing from research over the Higgs boson or dark matter to further measurements of the Standard Model parameters;

- **CMS, Compact Muon Solenoid:** it is the second general purpose detector at LHC. Together with ATLAS it shares the same scientific goals, although it uses different technical solutions and different magnet-system designs;

- **LHCb, Large Hadron Collider beauty:** this experiment is specialized in investigating the slight differences between matter and antimatter by studying quark beauty particles;

- **ALICE, A Large Ion Collider Experiment:** it is a heavy ion collision detector. It is designed to study the physics of strongly interacting matter at extreme energy densities, where the matter forms a new phase, called *quark-gluon plasma*.

1.3 The ATLAS detector

As stated above, the ATLAS experiment is a general purpose particle detector installed at LHC, which is used to study different kinds of physical phenomena. The detector is 44 m long [2], with an outer diameter of 25 m and weights approximatively 7000 t. ATLAS has a cylindrical symmetry, as well as the detectors of CMS and ALICE.

1.3.1 Coordinate System for ATLAS

The ATLAS detector describes the events using a coordinate system where the origin is set in nominal interaction point, with the z-axis defined by the beam direction and the x-y plane is transverse to it [3]. The positive x-axis is defined as pointing from the interaction

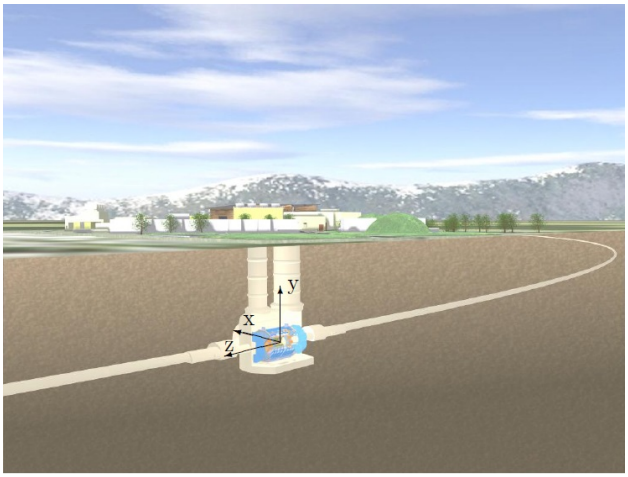


Figure 1.2: Coordinate system for ATLAS.

point to the center of the ring, while the y-axis is pointing upwards (see Figure 1.2). The azimuthal angle ϕ is around the beam axis and the polar angle θ is the angle from the beam axis. Pseudorapidity η is then defined as $\eta = -lntan(\frac{\theta}{2})$: its value ranges from 0, corresponding to the vector lying on the y-axis, to infinity where it is along the z-axis. Using the pseudorapidity, one can define the distance ΔR in the pseudorapidity-azimuthal angle space as $\Delta R = \sqrt{\Delta^2 R + \Delta^2 \phi}$. Other important variables, such as the transverse momentum p_T, the transverse energy E_T and the missing transverse energy E_T^{miss} are defined in the x-y plane.

1.4 ATLAS's structure

The structure of ATLAS's detector is presented in Figure 1.3. It is built with a cylindrical symmetry around the beam pipe axis centered on the interaction point, in order to guarantee a large acceptance in pseudorapidity (η). Ideally, it can be divided into four main regions: a barrel region, that covers low η phenomena; two end cap regions, with medium η; finally, two forward regions to cover the area with higher η. The overall detector is therefore made up of different groups of sub-detectors, designed to track proton-proton collisions and study the particles that come out from this system. As shown in Figure 1.3 the innermost of these sub-detector systems is called Pixel Detector; proceeding outwards calorimeters and muon spectrometers can be found. For what

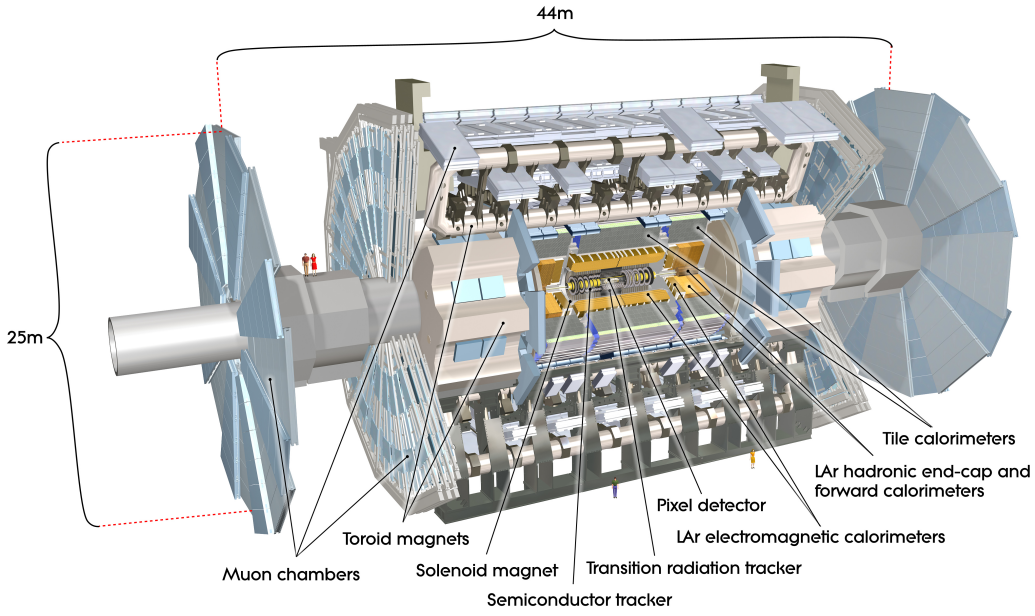


Figure 1.3: ATLAS detector overview.

concerns the magnetic field, a *central solenoid* surrounds the *Inner Detector* in order to provide a 2 T magnetic field, while being surrounded by toroids in barrel and end-caps sections of the muon spectrometer, providing magnetic fields of approximately 0.5 T and 1 T respectively. Particles produced from collisions of protons pass firstly through the *Inner Detector*, which covers the region with $|\eta| < 2.5$. The charged particles will interact with different layers of the detector and produce hits, which will be used to reconstruct their trajectory. The momenta and the charge of these particles can be measured, as their trajectories are bent by a 2 T magnetic field provided by the central solenoid. A global section view of ATLAS detector is provided in Figure 1.4, illustrating how and where different particles can interact. As the innermost layer of ATLAS, the Pixel Detector provides essential informations, such as the recognition of first and second vertices. Therefore, the Inner Detector is designed to have high granularity and high momentum measurement resolution. In order to achieve the performance requirements, semiconductor detectors are used for precise measurements close to the beam (*Pixel Detector* and *Semiconductor Tracker*), while a noble gas detector is used in the outer layer (*Transition Radiation Tracker*). Moving away from the collision point, the calorimeters

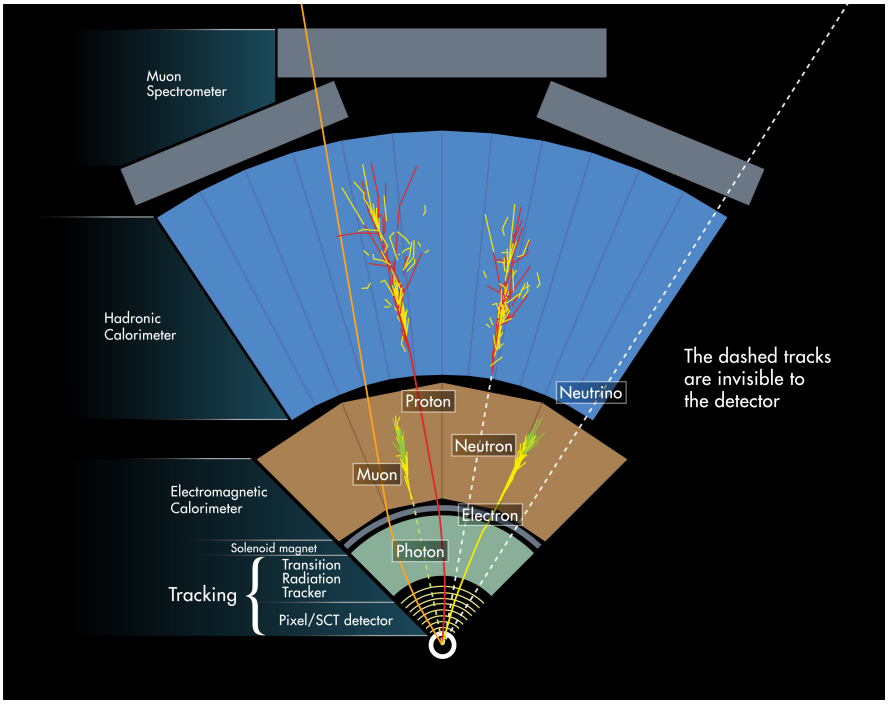


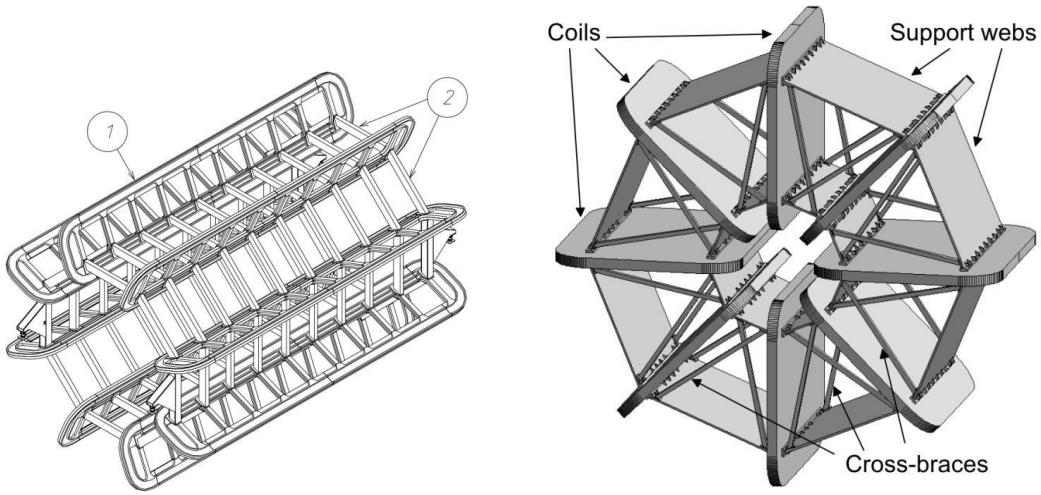
Figure 1.4: Section view of ATLAS detector in the transverse plane, illustrating layers' positioning.

can be found. They are composed of the hadronic calorimeters and the electromagnetic calorimeters, which are designed to detect hadrons or electron/photons respectively, as well as to measure their energy and coordinates. The incident particles can interact with the instrumentation via electromagnetic or strong process, producing so a shower of secondary particles.

The calorimeters will not stop muons, as they interact very little with the calorimeter absorber: therefore, the outermost layer of ATLAS, named *Muon Spectrometer*, is specifically designed to record and identify muons. Muons' spectrometer is made up of four different types of chambers: two types are intended to provide position and momentum measurement, while the other two types are meant to grant precise and robust information for the hardware-based trigger decision making.

1.4.1 Magnetic system

The ATLAS superconducting magnet system is shown in Figure 1.5.It is made up of a central solenoid, which provides magnetic field for the Inner Detector, surrounded by a system of three large air-core toroids, generating the magnetic field for the muon spectrometer. The overall dimensions of the magnetic system are 26 m in length and 20 m in diameter. At each end of the barrel toroid are inserted two end-cap toroids and lined up with the central solenoid. They have a length of 5 m, an outer diameter of 10.7 m and



(a) *Barrel toroid magnets for ATLAS.* (b) *End-Cap toroids magnets for ATLAS.*

Figure 1.5: Magnets system for ATLAS.

an inner bore of 1.65 m. The central solenoid extends over a length of 5.3 m and has a bore of 2.4 m. The unusual configuration of this magnetic system made its construction a real challenge, requiring a precise study and careful engineering. The Central Solenoid provides a central field of 2 T with a peak magnetic field of 2.6 T at the superconductor itself. The peak magnetic field on the superconductors in the Barrel Toroid and in the End-Cap Toroid are 3.9 T and 4.1 T respectively.
The position of the Central Solenoid in front of the calorimeter requires a careful minimization of the material in order to achieve the desired calorimeter performances. As a consequence, the Central Solenoid and the Liquid Argon calorimeters share one common vacuum vessel, thereby eliminating two vacuum walls.

1.4.2 Inner Detector

The Inner Detector (ID) is the closest detector to the beam pipe, therefore its design has to feature an excellent radiation hardness and a long term stability, while ensuring an adequate performance. As shown in Figure 1.6, the full Inner Detector is a cylinder

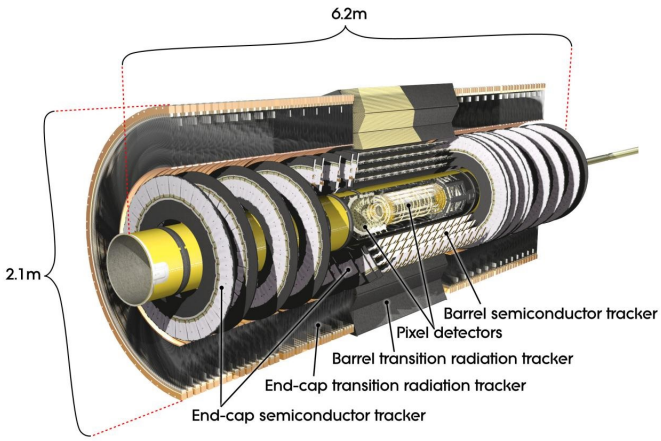


Figure 1.6: A section view of the ATLAS Inner Detector

with a length of 6.2 m and a diameter of 2.1 m. It covers the area with $|\eta| < 2.5$ and is segmented into cylindrical layers in the barrel region, while it has coaxial disks in the end-cap regions. As shown in Figure 1.7, ID is made up of three main different layers, which will be explained in the following paragraphs.

Transition Radiation Tracker

The Transition Radiation Tracker (TRT) detector is made up of thin drift tubes in the outer part of the ID, in which operates a non-flammable gas mixture composed by 70% Xe, 20% CO_2 and 10% CF_4, with a total volume of 3 m^3. These straws are placed along the pipe direction in the barrel region, while those in the end-cap region, have a radial direction.
Transition radiation is a form of electromagnetic radiation emitted when charged particle pass through an inhomogeneous mean, such as a boundary between two different means. The TRT occupies the largest space of the ID and provides the majority of the hits per track, hence highly contributing to the momentum measurement. TRT offers more hits per track in order to retrieve the information about momentum, even though it has a

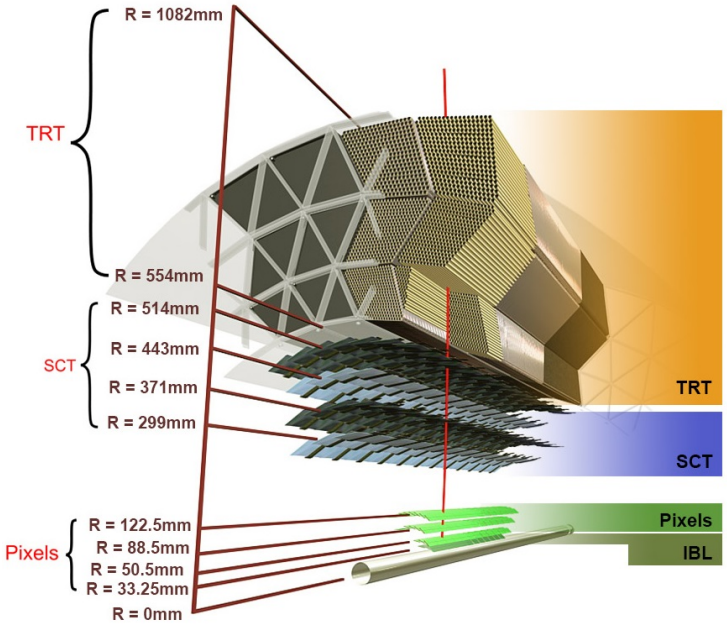


Figure 1.7: Structure and arrangement of the layers of Inner Detector in the barrel region.

lower precision compared to the silicon detectors. Unlike all other active parts of ID, the drift tubes do not need to be cooled and are not subject of radiation degradation, as the gas can simply be replaced.

Semiconductor Tracker

The SemiConductor Tracker (SCT) is a tracker made up of silicon strips, with a technology similar to the one employed in the Silicon Pixel detector. The reason for using such trackers, instead of pixels, are mainly economical, since the barrel of SCT covers more than 30 times the surface of Pixel detector (63 m^2 versus 1.73 m^2). The barrel part of the detector is made of 4 layers, while for the end-cap 9 layers for each side have been mounted.

Pixel Detector

The last and innermost detector is Pixel Detector, which is the most important. In order to have good vertex performance, this detector has been designed to have the finest granularity. The system actually consists of four cylindrical layers, named as the

following, going outwards: Insertable B-Layer, B-Layer, Layer 1 and Layer 2. In the next section it will be described in details.

1.5 Structure of Pixel Detector

The current configuration of Pixel Detector consists of four layers, as shown in the bottom of Figure 1.7. The three outer barrels, named B-Layer(L0), Layer 1 (L1) and Layer 2 (L2), were the first to be mounted, while the Insertable B-Layer (IBL), was the last, during 2013 LHC shutdown. Together, the L0, L1, L2 layers are composed by 112 long staves in total, each one being composed by 13 modules tilted on the z axis, by 1.1° towards the interaction point; furthermore, in order to allow overlapping, the staves are tilted by 20° on the x-y plane.
A module consists of many components, such sensors, 16 Front End (FE-I3) chips, a flex-hybrid, a Module Controller Chip (MCC) and a pigtail. FE-I3s are responsible for reading the charge signal from pixels. Each FE-I3 is $195\mu m$ thick, with a top surface of $1.09cm$ by $0.74cm$, counting 3.5 millions of transistors in $250nm$ CMOS technology. The module collects signals from the sensors, packaging them in a single data event, which is sent to the ROD board.

1.5.1 IBL

The Insertable Barrel Layer is a new pixel detector that has been inserted with a new shrunk beam-pipe inside the actual B-Layer. The reasons that led to this upgrade in 2013 shutdown are described hereafter.
Firstly, the inner layer was suffering a great mortality of pixels that would have increased over time, due to radiation exposure. This inefficiency caused a serious loss of b-tagging capability, which was completely restored thanks to IBL, even in the unfortunate event of B-Layer complete failure. Secondly, luminosity increase before the HL-LHC completion was too much for the read-out system and the pile-up would have led to readout inefficiencies. The higher occupancy induced by luminosity would have affected the B-Layer more than the other ones, leading to inefficiencies in b-tagging [4]. Thanks to IBL, the b-tagging capability was completely restored and some redundancy was added as well; furthermore, the high granularity led to lower occupancy and higher precision. Finally,

the tracking precision was strongly improved, thanks to IBL's pixels being closer to the interaction point. Improving the precision of impact parameters also resulted in a better sensitivity for signals in physics channels involving b-jets.

Being this close to the beam pipe forces some constraints that are not needed in other layers: electronics has to be much more radiation hard and the sensible area needs to cover more than the 70% of the surface, as in B-Layer. To achieve those objectives the FE-I4 was developed, leading to an active area of 90%.

Sensors for IBL

IBL's sensors are different from ATLAS's ones because of the technology chosen for the pixels. There were two main candidates:

- planar;
- 3D.

The main characteristics of these two technologies are explained hereafter, as well as the upgrade from FE-I3 chip to FE-I4.

Planar sensors were used in the B-Layer too, but the request on IBL's ones was much stricter: in fact, the inactive border had to pass from the older $1mm$ to the new $450\mu m$. Various studies were performed since B-Layer pixel were produced and now it is known that an irradiated sensor can collect much more charge if it is less thick. *3-D sensors* have a completely different geometry from planar ones and are read from two electrodes at once, since the charge collected by these pixels is low. Another downside of these sensors is that noise increases along with the number of electrodes and it's even affected by their diameter. Nevertheless, full 3-D sensors' active area extends much closer to the surface, reducing non-sensible volume. The faces of 3-D sensors, independently from the type, are much closer one another, allowing a much lower bias voltage (150V versus 1000V of a planar sensor). This also leads to a lower leakage current and thus less cooling. When a particle passes through the electrode area, efficiency results diminished by 3.3%. This effect affects only in perpendicular particles and thus will not affect IBL for its sensors are tilted by 20°.

FE-I4

FE-I4, shown in Figure 1.8 is the new ATLAS pixel chip developed to be used in upgraded luminosity environments, in the framework of the Insertable BLayer (IBL) project but also for the outer pixel layers of Super-LHC.

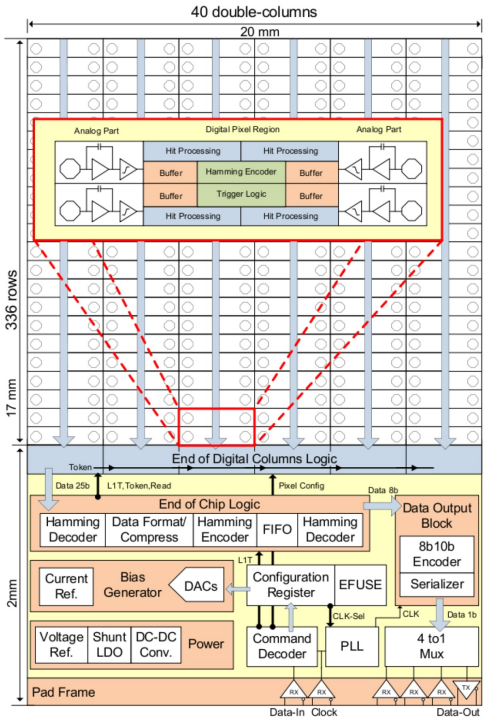


Figure 1.8: FE-I4 Layout

FE-I4 is designed in a 130 nm CMOS process, in an 8 metal option with 2 thick aluminium top layers for enhanced power routing. Particular care has been taken to separate analog and digital power nets. With the thinning down of the gate oxide, the 130 nm CMOS process shows an increased radiation tolerance with respect to previous larger feature size processes.

The motivations for the redesign of the pixel Front-End FE-I3 came from several aspects, related to system issues and physics performances of the pixel detector. With a smaller innermost layer radius for the IBL project and an increased luminosity, the hit rate increases to levels which the FE-I3 architecture is not capable of handling.

In particular, it was shown that the current FE-I3 column-drain architecture scales badly with high hit rates and increased FE area, leading to unacceptable inefficiencies for the

IBL. FE-I4 stores hits locally to avoid a column-drain based transfer. The FE-I4 pixel size is also reduced, from $50 \times 400\mu m^2$ to $50 \times 250\mu m^2$ which reduces the pixel cross-section and enhances the single point resolution in z direction. FE-I4 is built up from an array of 80 by 336 pixels, each pixel being subdivided into analog and digital section. The total FE-I4 active size is 20 mm (z direction) by 16.8 mm (ϕ direction), with about 2 mm more foreseen for periphery, leading to an active area of close to 90% of the total. The FE is now a standalone unit avoiding the extra steering of a Module Controller Chip for communication and data output. Communication and output blocks are included in the periphery of the FE. Going to a bigger FE size is beneficial with respect to active over total area ratio as well as for the building up of modules and staves. This leads to more integrated stave and barrel concepts, and as a consequence reduces the amount of material needed per detector layer. Such a reduction of material has a drastic effect on physics performance, e.g. on b-tagging efficiency vs. light quark rejection factor. One of the main advantages of having a big FE is also the cost reduction.

Chapter 2

Off detector electronics

Before explaining the reasons that led to the development of a new board, it is wiser to look at the current set-up of the off-detector electronics for the Insertable Barrel Layer. High-energy physics experiments usually distinguish between *on-detector* and *off-detector* electronics: the former refers to the front-end electronics implemented near the detector itself, where radiation resistance is a fundamental parameter for the electronics; the latter refers to the readout system that is implemented far from the detector, thereby removing the requirement of radiation resistance and allowing the employment of more powerful devices.

The following section will present the off-detector electronics topic into details, in order to give a precise environment for which the new Pixel-ROD board was conceived and to understand the requirements that it needs to fulfill.

2.1 IBL's electronics

The readout system for IBL requires an appropriate off-detector apparatus, which is schematically shown hereafter, in Figure 2.1. This readout system is made of several components:

- Back of Crate (BOC) board:
  - Optical modules to interface FE-I4 chips with BOC board;
  - S-Link for sending data from the BOC board to the ATLAS TDAQ system.

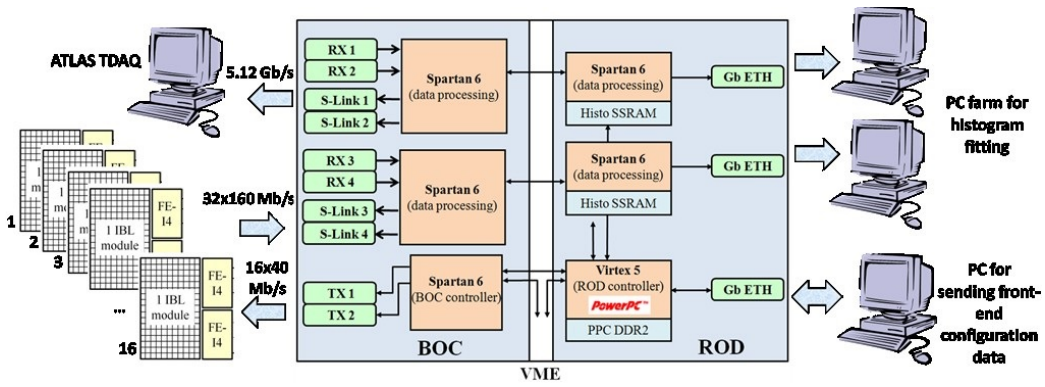


Figure 2.1: Schematic block view of th IBL readout system.

- Read Out Driver (ROD) board:
  - Gigabit Ethernet to send front-end calibration's histograms.
- VME Crate;
- TTC Interface Module (TIM);
- Single Board Computer (SBC).

Each BOC-ROD pair is able to interface and route data coming from 16 IBL modules (32 FE-I4 chips, for a total input bandwidth of 5.12 Gb/s), meaning that the whole IBL readout requires 15 BOC-ROD pairs, that can be all placed in a single VME crate.

A fraction of this set-up is implemented here in Bologna as well, where a VME crate hosts and interfaces a few BOC boards with RODs; furthermore, few front-end chips are available as well, even though the optical links are missing, mainly because of their cost. The data path is presented in Figure 2.2 and is quite simple: the 32 front-end chips FE-I4 drive 32 serial lines, each sup-

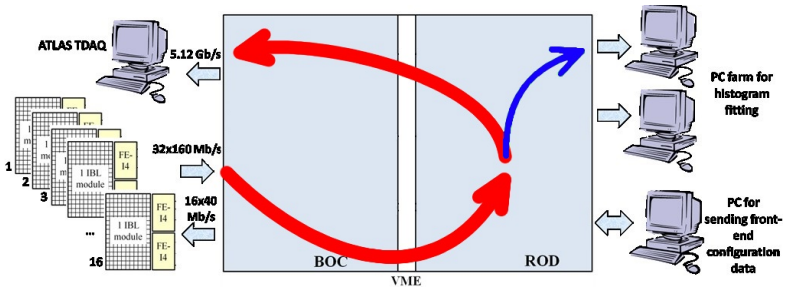


Figure 2.2: Complete visual layout of the data acquisition system. In red the normal data path, in blue the deviation of the Histogram data path from the normal one.

porting 160 Mb/s, connected to the BOC board via optical links. Here the signal from each line is converted from optical to electrical, then demultiplexed to one 12-bit-wide bus, which proceeds towards the ROD board, through the VME backplane connector. The ROD board begins the real data formatting, to build up a *ROD data frame*, that is sent to the TDAQ computers. On the ROD board, data can take two different paths (see Figure 2.2): the first routes the ROD data events back to the BOC, where four S-Link modules process the data towards the ATLAS TDAQ PCs, implementing a total output bandwidth of 5.12 Gb/s; the second route for data is toward the PC for histogram making, exclusively used during calibration runs in order to properly calibrate the FE-I4 chips.

2.1.1 IBL BOC

The BOC board, shown in Figure 2.3, is responsible for handling the control interface to the detector, as well as the data interface from the detector itself. Also, one major

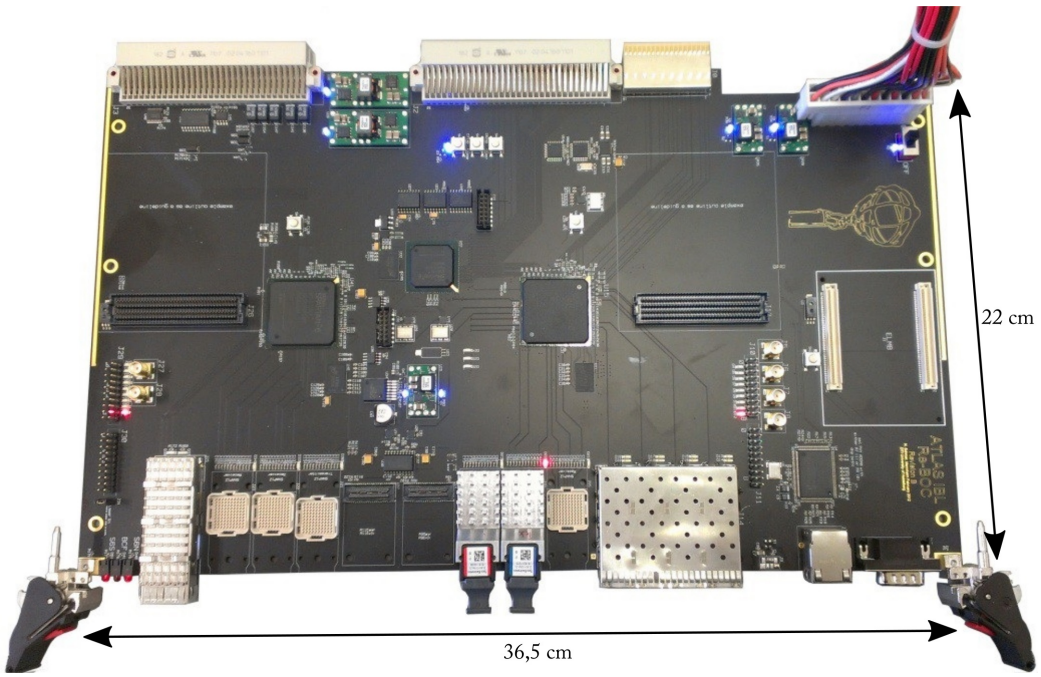


Figure 2.3: IBL BOC board

task of the BOC is to provide the clock to the front end chips connected. In order to do so, the clock is received by the BOC from the TIM and can be delayed, if needed.

Furthermore, a Phase Locked Loop (PLL) generates copies of this clock for the ROD and the detector. The detector clock is then handled by the FPGAs and coded into the control streams for the individual detector modules.
The IBL BOC contains three Xilinx Spartan FPGAs:

- one BOC Control FPGA (BCF);
- two BOC Main FPGAs (BMF).

BOC Control FPGA

The BOC Control FPGA, as the name suggests, is responsible for controlling the FPGA. A Microblaze embedded processor is instantiated on this FPGA, mainly in order to manage the Ethernet connection for the board, but it is able to implement some self test for the board as well. The BCF is also responsible for FPGAs configuration, where a two-step start-up sequence is used. Firstly, the BCF loads its configuration in "Master Serial Peripheral Interface" mode from a 64 Mbit SPI FLASH. Subsequently, the BCF firmware reads the configuration data for the two main FPGAs from a second SPI FLASH and downloads it via the Slave Serial configuration ports. Depending on the configuration, the BCF can also load software from a third SPI FLASH.

BOC Main FPGA

The two BMFs are responsible for encoding the configuration data for FE-I4 front end chips, which are coming from the ROD, into a 40 Mb/s serial stream that is sent straight to the FE-I4. This configuration stream itself can be generated by the BMFs, mainly for testing purposes. This two FPGAs also manage the deserialization of the incoming data from front-end chips on the RX path; after the data collection and the word alignment, the decoded data are sent to the ROD board. On the TX side, these two Spartan FPGAs also manage the optical connection via four S-Links to the ATLAS TDAQ system.

2.1.2 IBL ROD

The Insertable Barrel Layer Read Out Driver (IBL ROD) [5] is a board meant to be the upgrade of the older ATLAS Silicon Read Out Driver (SiROD), that is used for the ATLAS Off-Detector electronics sub-system in order to interface with Silicon Tracker

(SCT) and Pixel B0, L1 and L2 Front End Detector modules.
This board is designed in order to accomplish some tasks, like propagating timing and trigger signals to the front-end electronics, as well as sending an appropriate configuration to them. Indeed, the most important task for the ROD is accomplished during physical runs, when the board receives data and event fragments from the 32 FE-I4 chips and transform them into a *ROD data frame*, which is sent back to the ATLAS TDAQ, through the BOC's S-Link connections (see Figure 2.2).

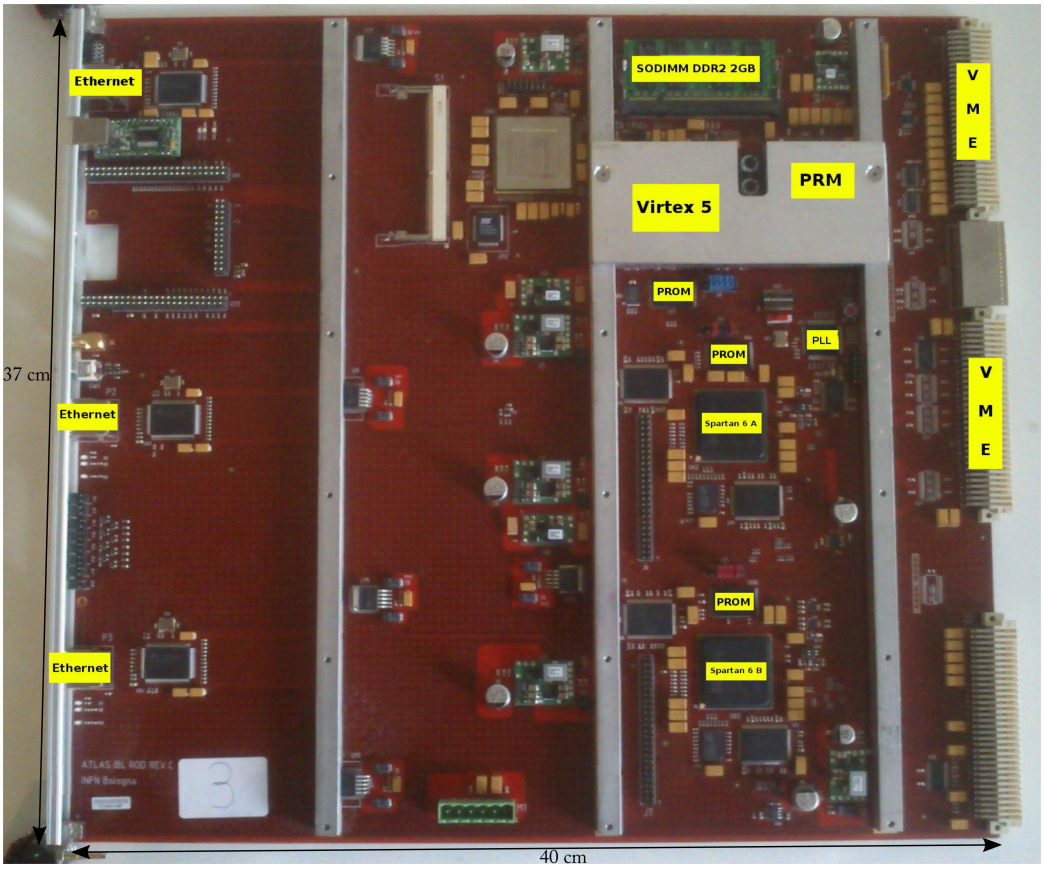


Figure 2.4: The IBL ROD board

The IBL ROD (shown in Figure 2.4) is a 14-layers PCB on with many components, hereafter reported:

- One Digital Signal Processor MDSP (Texas Instruments TMS320C6201 - GJC200), which is currently not used;
- One Program Reset Manager (PRM) FPGA (Xilinx Spartan-6 XC6SLX45-FGG484);

- One ROD Controller (Master) FPGA (Xilinx Virtex-5 XC5VFX70T-FF1136);
- Two "Slave" FPGAs (Xilinx Spartan-6 XC6SLX150-FGG900);
- One Phase-Locked Loop PLL (Lattice ispClock 5620);
- 32 MByte SDRAM DDR;
- Four Mbit FLASH SST39VF040-70-4C-NH;
- Two GByte DDR2 SODIMM;
- 64 Mbit FLASH Atmel AT45DB642D;
- Three Gbit Ethernet interfaces with PHY DP83865.

Hereafter the main devices of the ROD board are described, in order to present the tasks and functionalities of each of them.

ROD Master

The Virtex-5 FPGA is the Master of the Read Out Driver, which must interface with the front-end chips, the triggers that come from TTC Module and all the information that refers to the Trigger itself. This FPGA contains a PowerPC, an embedded hard processor with its specific architecture. The tasks of this FPGA are many, like processing the trigger information and deliver it to the Spartan FPGAs or sending the event information (Event ID, Trigger Type and Bunch Crossing ID) to both Spartan FPGAs, so that they can be written in the header of the ROD event.

ROD Slaves

Both Spartan-6 FPGAs work as slaves on the ROD board and implement an embedded soft processor, named Microblaze, with its specific architecture. All data generated by IBL during ATLAS experiments pass through these two FPGAs and are collected inside the SSRAM; moreover, during calibration runs histograms can be generated and sent to the histogram server. Furthermore, each Spartan manages its Ethernet connection, through which histogram can be sent, as stated above.

Lattice PLL

Lattice ispClock 5620 Phase-Locked Loop (PLL) is an integrated circuit, that generates an output clock with a fixed frequency and phase relationship with the input clock. The function of the PLL is to compare the distributed clock to the incoming reference clock and change the phase and frequency of its output until the reference and feedback clocks are matched in phase and frequency. For what concerns clock sources, this board receives the clock (with a frequency of 40 MHz) from the BOC during standard runs; indeed, the need to test the ROD in a standalone mode required an internal quartz clock generator on the board. It is possible to choose between these two different clock sources by controlling a dedicated DIP switch. After that, the PLL multiplies the 40 MHz input clock, generating one with a frequency of 100 MHz. The 100 MHz and the 40 MHz clocks are then distributed all over the board.

2.1.3 TIM

The TTC (Timing, Trigger and Control) Interface Module (TIM) interfaces the ATLAS Level-1 Trigger system signals to the pixel Read-Out Drivers using the LHC-standard TTC and Busy system. This board is designed to propagate the TTC clock all over the experiment: for what concerns the IBL off-detector electronics, the TIM sends the clock to the BOC board, which then propagates it to the ROD, as stated above. Furthermore, the TIM receives and propagates triggers through the custom backplane (P3 connector).

2.1.4 SBC

The Single Board Computer, as the name suggests, is actually a computer mounted on a 6U board with a VME interface chip. It is used to control all the VME operations on the ROD and it can actually program the ROD FPGAs, usually after power up. It can also be used to monitor the temperature, or voltages, on the RODs master device.

2.2 The road towards Pixel-ROD

Since the ROD board described in the previous section met all the expectations that ATLAS experiment had, it was decided to implement this system also for all other layers

(B0, L1, L2). The effort on the IBL electronics required 15 ROD boards to be produced and tested in 2014, while the remaining layers required about 110 boards, whose last batch has just been installed at ATLAS. Alongside with this work, a huge effort was also spent into the development and upgrade of the firmware for the ROD boards.
The knowledge acquired with IBL electronics made also clear the limits of this system, especially looking to the future upgrade of the whole LHC detector to the higher luminosity HL-LHC. The established Long Shutdown in 2023 [6] for the upgrade of the whole LHC will bring the nominal luminosity from five to seven times the actual one, with an expected luminosity peak of $L_{peak} = 7.5 \times 10^{34} cm^{-2} s^{-1}$. Such huge improvement in luminosity also means that the electronics will need to withstand a much higher data rate.

Looking into this direction, many electronic boards have been presented for the readout of such experiments [7]. All the available electronic board's projects share a common feature: the implementation of an electronic board designed to be flexible and highly configurable, with PCIe interface, as well as powerful FPGAs connecting to many optical transceivers.

Looking into high-speed devices, we decided to keep working with FPGAs from Xilinx, upgrading to the 7-Series family. This decision was taken in order not to waste all the experience and the efforts spent on the ROD board, while allowing the portability of the firmware onto this new one, named *Pixel-ROD*, after upgrading it on the newly introduced platform. Furthermore, given the success of the Master-Slave architecture from the ROD board, as well as for the reason just explained above, it was decided to use two FPGAs on the Pixel-ROD board. Since the process of creating and debugging such highly complex boards often turns out to be very time consuming, it was decided to design the Pixel-ROD from two evaluation boards made by Xilinx: the KC705 and the ZC702.

2.2.1 KC705

For the *slave* device we looked for a powerful FPGA from Xilinx's Kintex family, a good example of which was already given by a Xilinx's evaluation board, named KC705. The KC705, shown in Figure 2.5, was an interesting board for us in many ways, whose main devices and features are listed below [8]:

Figure 2.5: Xilinx's KC705 demo board

- Kintex-7 28nm FPGA (XC7K325T-2FFG900C);
- 1GB DDR3 memory SODIMM;
- PCI Express Gen2 8-lane endpoint connectivity;
- SFP+ connector;
- Two VITA 57.1 FMC Connectors (one HPC, one LPC);
- 10/100/1000 tri-speed Ethernet with Marvell Alaska 88E1111 PHY;
- 128 MB Linear Byte Peripheral Interface (BPI) flash memory;
- 128 Mb Quad Serial Peripheral Interface (SPI) flash memory;
- USB-to-UART bridge;
- USB JTAG via Digilent module;
- Fixed 200 MHz LVDS oscillator;
- I^2C programmable LVDS oscillator;

First of all, the KC705 is a PCIe board: this interface is very important for newly developed boards, since it is the one that is more likely to replace the role of slower VME buses. In fact, VME bus data rate is 200 MB/s, while an 8 lanes PCIe Gen2 bus supports a data rate of 4GB/s. Furthermore, PCIe allows a new possible installation configuration: one or two of these boards can be directly connected on the motherboard of TDAQ PCs, thereby giving straight access to the main resources of the PC, like CPU and RAM, for a faster response and providing also an easier installation. This configuration is the one towards which ATLAS, as well as other experiments like CMS, are aiming to for the experimental phase that will start after the Long Shutdown in 2023.

Kintex-7 FPGA

The new Kintex-7 is a powerful medium-range FPGA, that can easily fulfill the role of both Spartan-6 devices on the ROD board [9, 10].

The Xilinx Kintex-7 XC7K325T-2FFG900 on this board has the following features:

- Advanced high-performance FPGA logic based on real 6-input lookup table (LUT) technology configurable as distributed memory;
- High-performance DDR3 interface supporting up to 1866 Mb/s;
- High-speed serial connectivity with built-in 16 gigabit transceivers (GTX) from 600 Mb/s to maximum rates of 12.5 Gb/s, offering a special low-power mode, optimized for chip-to-chip interfaces;
- A user configurable analog interface (XADC), incorporating dual 12-bit analog-to-digital converters (ADC) with on-chip temperature and supply sensors;
- Powerful clock management tiles (CMT), combining phase-locked loop (PLL) and mixed-mode clock manager (MMCM) blocks for high precision and low jitter;
- Integrated block for PCI Express (PCIe), for up to x8 Gen2 Endpoint and Root Port designs;
- 500 maximum user I/Os (excluding GTX) and 16Kb of Block RAM (BRAM).

	Kintex-7 (Pixel-ROD)	Spartan-6 (ROD)
Logic cells	326080	147443
BRAM Blocks (18 Kb)	890	268
Memory Interface	1866 Mb/s	800 Mb/s

Table 2.1: Brief comparison between Spartan-6 and Kintex-7 features.

VITA 57.1 FMC connectors

Another strong point of this board are its two VITA 57.1 FMC [11] connectors, that allow to interface many external devices, thanks to this particular and standardized connection, shown in Figure 2.6. This additional I/O connection is achieved thanks to external *I/O*

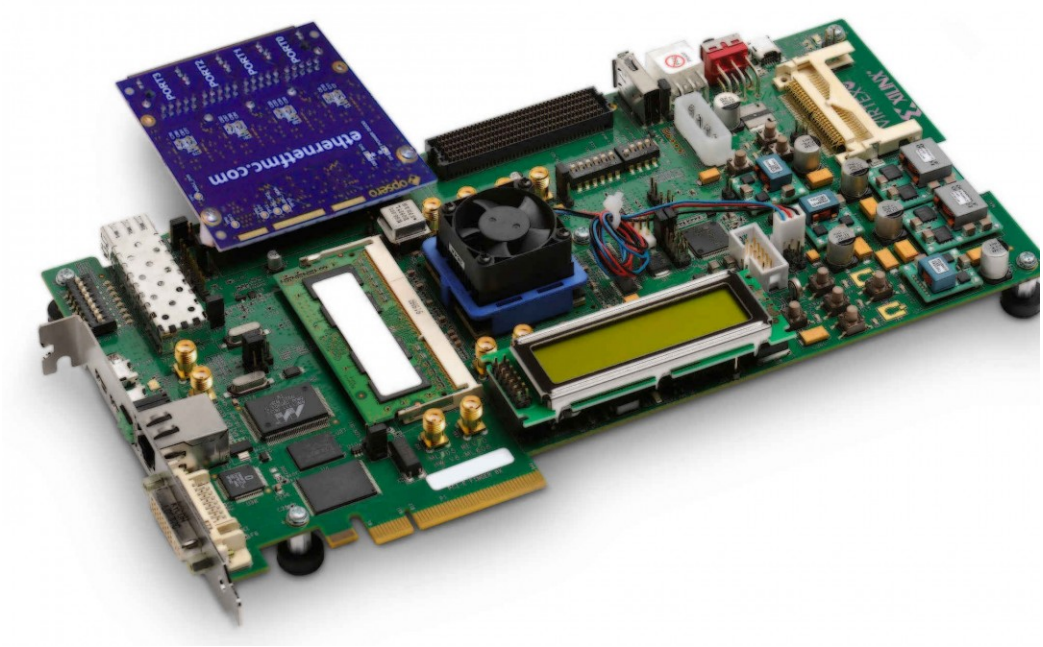


Figure 2.6: A Xilinx demo board displayed as carrier, with external FMC mezzanine.

mezzanine module. In this standard, the board carrying the FPGA acts as *carrier* for the FMC mezzanine module, which enables the board to have flexible and different external connections. This connector standard supports up to 10 Gb/s transmission with adaptive equalized I/O, as well as single ended and differential signalling up to 2 Gb/s.
Two types of FMC connectors exist, both of which are present in the KC705 board (see Figure 2.5):

- Low Pin Count (LPC) connector, providing 68 user-defined single-ended signals or 34 user-defined differential pairs, one transceiver pair, one GTX clock and one differential clock (160 total pins);
- High Pin Count (HPC) connector, providing 160 user-defined single-ended signals (or 80 user-defined differential pairs), 10 serial transceiver pairs, two GTX clocks, four differential clocks (400 total pins).

On the KC705 four GTX transceivers are wired to the FMC HPC connector, while only one is connected to the FMC LPC. Therefore these connectors allow a new level of flexibility for FPGA boards, granting the user the possibility to properly choose its I/O devices by changing mezzanine module. Furthermore, this standard eliminates the need for complex protocol logic on the mezzanine, since devices placed there can be connected to the main board JTAG and I2C chains, therefore allowing a lower complexity of FPGA design and a simpler debug.

PCI Express connector

Since the Kintex-7 FPGA on KC705 board supports a PCIe connection up to Gen2 x8, the 8-lane PCI Express [12] edge connector performs data transfers at the rate of 5 GT/s. The PCIe transmit and receive signal traces have a characteristic impedance of 85Ω ±10%. The PCIe clock is routed as a 100Ω differential pair.

2.2.2 ZC702

The second demo board we took inspiration from is the Xilinx's ZC702, shown in Figure 2.7. As previously described, the ROD board hosts a Virtex-5 FPGA, implementing an embedded hard processor named *PowerPC*. This feature is highly important on the master FPGA, since it allows to write software (using C or C++ language), that can be run on this processor. For this reason the ZC702 becomes interesting for our project, since its FPGA embeds a hard processor [13], which allows to avoid changes in the firmware each time a particular function needs to be tested.

Hereafter the main features of the ZC702 demo board are listed:

- Zynq 7000 FPGA (XC7Z020-1CLG484C), featuring two ARM Cortex-A9 MPCore hard processors;

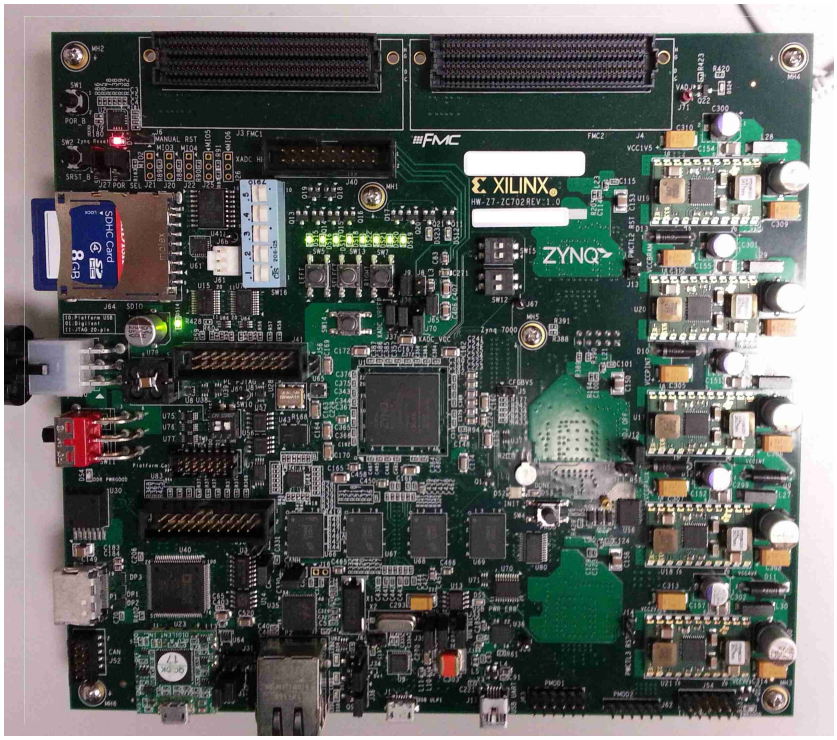


Figure 2.7: Top view of ZC702 demo board

- 1 GB DDR3 component memory (Micron MT41J256M8HX-15E);
- 10/100/1000 tri-speed Ethernet with Marvell Alaska 88E1116R PHY;
- 128 Mb Quad SPI flash memory;
- USB-to-UART bridge;
- USB JTAG interface using a Digilent module;
- Two VITA 57.1 FMC LPC connectors;
- Fixed 200 MHz LVDS oscillator;

Zynq-7000 FPGA

The Zynq 7000 FPGA consists of an integrated Processor System (PS) and Programmable Logic (PL). The PS integrates two ARM®Cortex™-A9 MPCore™ with a frequency up to 667 MHz, AMBA™ interconnect, internal memories, external memory interfaces,

and peripherals including USB, Ethernet, SPI, SD/SDIO, I2C, CAN, UART, and GPIO. The PS runs independently of the PL and boots at power-up or reset.

Even though a detailed description of the architecture of the Zynq is not in the intention of this thesis, it necessary to proceed with a brief explanation of this system, in order to provide a better understanding of the topics covered in the next chapter.
A schematic diagram of the Zynq FPGA is given in Figure 2.8, which illustrates the functional blocks. The Processor System is surrounded by a blue line in Figure 2.8,

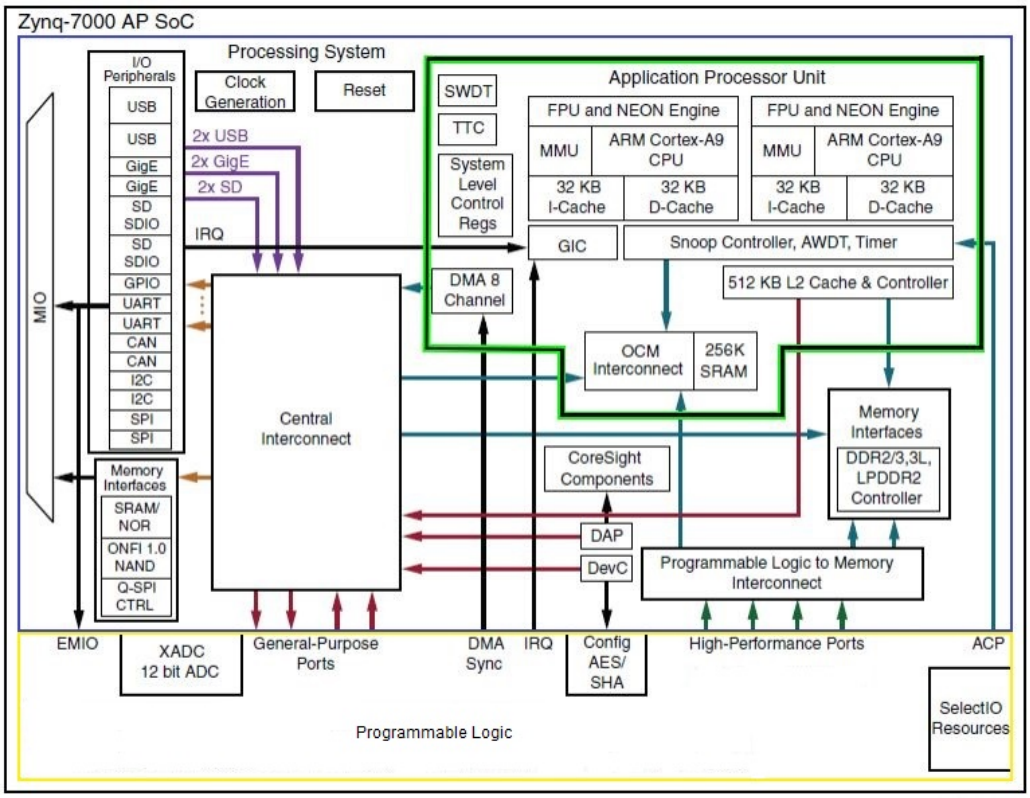


Figure 2.8: Zynq 7000 architecture overview

while the Programmable Logic by a yellow line. The PL part is structured just as any other FPGA, featuring 85K logic cells and 560 KB of Block RAM, making it equivalent to an Artix®-7 FPGA from the Xilinx 7 Series family.
Indeed, the Processor System is far more complex, as figure Figure 2.8 shows. It can be divided in four main functional blocks:

- Application Processor Unit (APU), within the green line;

- Memory interfaces;
- I/O peripherals (IOP);
- Central interconnect.

APU The Application Processor Unit [14], shown in Figure 2.9, is mainly made up of two ARM processing cores, each one with its associated computational units: a NEON Media Processing Engine (MPE) and Floating Point Unit (FPU), a Memory Management Unit (MMU) and a Level 1 cache memory, splitted in two sections for instructions and data.

The APU also contains a Level 2 cache memory, and a further On Chip Memory (OCM). Finally, a Snoop Control Unit (SCU) forms a bridge between the ARM cores and the Level 2 cache and OCM memories.

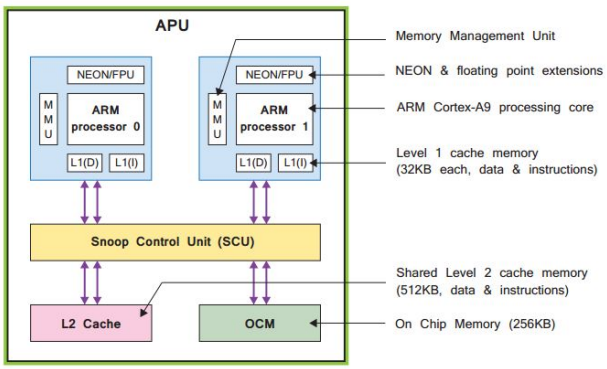


Figure 2.9: Simplified block diagram of the Application Processing Unit

The ARM Cortex-A9 processor mounted on this demo board can be clocked up to 667 MHz. Each of the two cores has separate Level 1 caches for data and instructions, both of which are 32 KB; as in the general case, this permits local storage of frequently required data and instructions for fast access times and optimal processor performance.The two cores additionally share a larger Level 2 cache of 512 KB for instructions and data, and there is a further 256 KB of on-chip memory within the APU. Since modern computing systems are always very demanding for what concerns memory requirements, it often happens that the finite amount of memory available is not enough. In this respect, *virtual memory* can prove to be very useful, as it makes appear as there is more memory available in the system than what actually is. Here is where the Memory Management Unit (MMU) becomes helpful, since its role is to translate between physical and virtual addresses.

The Snoop Control Unit (SCU) undertakes several tasks, involving the interface between the processor and Level 1 and 2 cache memories and is mainly responsible for maintain-

ing memory coherency between the processor data cache memories, which are marked as L1(D) in Figure 2.9, and the shared Level 2 cache memory. It also initiates and controls access to the Level 2 cache, arbitrating between requests from the two cores where necessary.

From a programming perspective, support for ARM instructions is provided via the Xilinx *Software Development Kit* (SDK) which includes all necessary components to develop software for deployment on the ARM processor, as it will be shown more accurately in the next chapter.

Finally, one last additional functionality to the main ARM processor is the NEON engine, which provides Single Instruction Multiple Data (SIMD) facility to enable acceleration of DSP type algorithms. NEON instructions are an extension to the standard ARM instruction set, and can either be used explicitly, or by ensuring that the C code follows an expected form and thus allows NEON operations to be inferred by the compiler. As the SIMD term suggests, the NEON engine can accept multiple sets of input vectors, upon which the same operation is performed simultaneously to provide a corresponding set of output vectors. This style of computation is suitable for applications involving digital signal processing (DSP), which operate on a large number of data samples simultaneously with algorithms that are inherently parallel.

Memory interfaces The Zynq has two main memory interfaces, which connect it to few different memory types. Through the Central Interconnect, the APU can access the controllers of QSPI and other built in memories. However, the APU can access another very important memory controller, which is the DDR memory controller.
The DDR memory controller supports DDR2, DDR3, DDR3L, and LPDDR2 [15] devices and consists of three major blocks: an AXI memory port interface (DDRI), a core controller with transaction scheduler (DDRC) and a controller with digital PHY (DDRP). The DDR interface (DDRI) arbitrates the requests from the eight ports (four reads and four writes). The arbiter selects a request and passes it to the DDR controller and transaction scheduler (DDRC). The arbitration is based on a combination of how long the request has been waiting, the urgency of the request, and if the request is within the same page as the previous request. The DDRC receives requests from the DDRI through a single interface. Both reads and writes flow through this interface. Read requests include a tag field that is returned with the data from the DDR. The PHY processes read/write

requests from the controller and translates them into specific signals within the timing constraints of the target DDR memory. Signals from the controller are used by the PHY to produce internal signals that connect to the pins via the digital PHYs. The DDR pins connect directly to the DDR devices via the PCB signal traces. The DDR controller PHY (DDRP) drives the DDR transactions.

I/O Peripherals The I/O peripherals (IOP) are a collection of industry-standard interfaces for external data communication. The Zynq PS features a variety of interfaces, which can be ideally divided into 2 groups: PS-PL interfaces and PS to External Components interfaces. The latter and most important group of interfaces is shown on the left side of Figure 2.8, where connection between the PS and external interfaces is achieved primarily via the Multiplexed Input/Output (MIO), that provides 54 pins of flexible connectivity, meaning that the mapping between peripherals and pins can be defined as required. Certain connections can also be made via the Extended MIO (EMIO), which is not a direct path from the PS to external connections, but instead passes through and shares the I/O resources of the PL. The available I/O includes standard communications interfaces, like Gigabit Ethernet Controllers, USB Controllers, UART Controllers and General Purpose Input/Output (GPIO), which can be used for a variety of purposes, including simple buttons, switches, and LEDs.

Interconnects The interconnects that are located within the PS are designed to work as switches with the specific task to facilitate the communication of read, write and response transactions between master and slave client. There are many interconnects within the Zynq PS, as shown in Figure 2.8: the most important one is the Central Interconnect, which interfaces many peripherals with the APU, therefore allowing the latter to send data as well as register configuration for the I/O peripheral interfaces. Other interconnects can be seen in Figure 2.8, such as PL to Memory Interconnect or OCM Interconnect, all sharing the same task, that is to allow communication exclusively from the permitted masters to specific slaves, via AXI protocol communication.

2.3 The Pixel-ROD board

The analysis of the two demo boards, described in the previous section, was fundamental for the realization of the new Pixel-ROD board. Coming this far, we decided to design the Pixel-ROD as a fusion from both KC705 and ZC702. While creating Pixel-ROD by merging the two demo boards, many features had to be removed, since not useful for a readout board, as we wanted the Pixel-ROD to be. Furthermore, other features had to be re-designed, as they needed to be shared among the whole hardware of the new board.

Speaking about KC705, with reference to the Figure 2.5, the main features that had been removed were of course the LCD display, the SD card reader and the HDMI port, as well as a few GPIO buttons and LEDs. On the side of ZC702, almost the same features were removed: SD card reader, HDMI port, GPIO buttons and LEDs were removed, along with one of the two LPC FMC, the USB port and PMODS connectors. All these features removed made room to more useful ones, like buses between FPGAs, which are mandatory since they are needed to implement a ROD-like "Master-Slave" architecture.

Therefore, principal devices and features that are implemented on the Pixel-ROD board are the following:

- Kintex-7 28nm FPGA (XC7K325T-2FFG900C);
- Zynq 7000 FPGA (XC7Z020-1CLG484C), featuring two ARM Cortex-A9 MPCore;
- 2 GB DDR3 memory SODIMM (Kintex DDR);
- 1 GB DDR3 component memory (Micron MT41J256M8HX-15E, Zynq DDR3);
- PCI Express Gen2 8-lane endpoint connectivity;
- SFP+ connector;
- Three VITA 57.1 FMC Connectors (one HPC, two LPC);
- Two 10/100/1000 tri-speed Ethernet with Marvell Alaska PHY;
- Two 128 Mb Quad SPI flash memory;
- Two USB-to-UART bridges;

- USB JTAG interface (using a Digilent module or header connection);
- Two fixed 200 MHz LVDS oscillators;
- I^2C programmable LVDS oscillator;

As previously stated, on the Pixel-ROD a bus has been added between the two FPGAs in order to obtain the necessary communication. There are three main types of buses, each one intended to provide specific features: the first and most important type of connection is given by the 21-bit differential bus, intended to provide an high speed communication link between Kintex and Zynq; the second type of bus is a 1-bit differential line whose main task is to share a common clock between the two FPGAs, whenever is needed; the last type of interconnection bus is a 5-bit wide, single ended bus, which is a general purpose bus. Along with these features that have been newly introduced, others had to

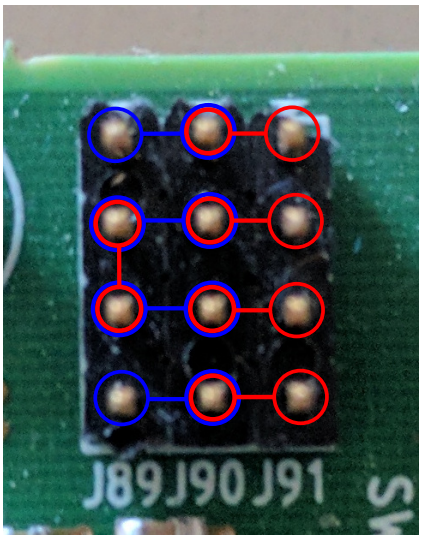


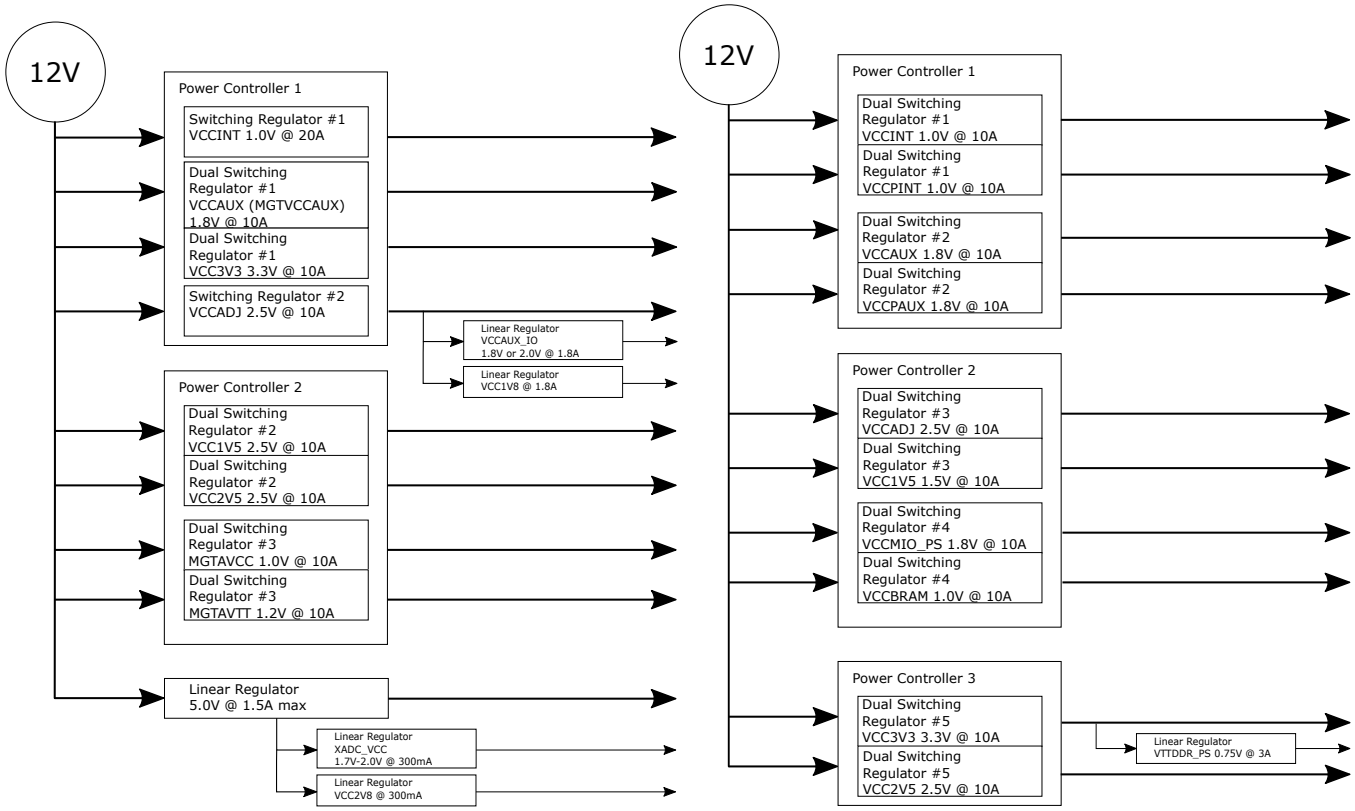
Figure 2.10: Custom JTAG configuration header. The blue lines and circles highlight the *full JTAG chain* configuration, while in red it is highlighted the one activating the internal JTAG, excluding the Kintex from the global chain.

be tuned to fulfill their roles, like JTAG chain and power supply stage. The former has been modified in order to include the main devices on the board, which are of course both FPGAs and two out of three FMC connectors (those controlled by Kintex). The JTAG chain was modified also in another way: a 12 (3x4) pin header (see Figure 2.10) was added in order to allow the possibility to exclude the Kintex from the JTAG chain. This

feature was added with the idea of prevent unwanted programming of the slave FPGA. Furthermore, an internal JTAG from Zynq to Kintex has been added: this feature allows to program the slave FPGA with the desired firmware, using Zynq as Master, once its firmware is defined. This can be very helpful during debugging session, when the Pixel-ROD board can be installed inside a PC, where it can be difficult to access to a JTAG port.

The last feature that had to be re-invented while designing Pixel-ROD board is the power supply stage. Coming from the two demo boards, which have the same devices for the power supply stage, we decided not to barely copy it, but to try to merge them instead. The reason behind this decision is mainly one: since the Pixel-ROD is intended to fit into a PC case [16], replying both supply stages would have been faster, but would have compromised the possibility to insert the board in a PC case for space related reasons. In Figure 2.11 it is shown a schematic representation of both demo boards power supply stage. As the figures show, they are very similar, since they both take in 12V from a Molex connector and provide this to different power controllers. These power controllers cover a very important role, since they do not only obtain from the 12 V input the desired output voltages, but they provide the precise sequence of power up. With the growth in FPGA complexity, the need for a precise sequence of power up began to play a very relevant role. On the Pixel-ROD board this task is achieved thanks to three *Digital Pulse-Width Modulation (PWM) System Controller* (UCD9248) [17], the same devices implemented on the demo boards. The UCD9248 is a 4 rail, 8 phases synchronous buck digital PWM controller designed for non-isolated DC/DC power application. When multiple power stages are configured to drive a voltage rail, the UCD9248 automatically distributes the phase of each DPWM output, in order to minimize ripple. This device integrates dedicated circuitry for DC/DC loop management with RAM and flash memory and a serial interface to support configurability, monitoring and management. In order to facilitate the configuration of these devices, a PC based Graphical User Interface (GUI), named Fusion Digital Power Designer, is provided by Texas Instrument. This tool allows to configure the system operating parameters for the application, storing the configuration to on-chip non-volatile memory. Furthermore, it is possible to get real time measurements of sensed voltages, once the device in configured and correctly interfaced.

As it will be described in more details in the next chapter, these devices can be



(a) *KC705 power supply stage.* (b) *ZC702 power supply stage.*

Figure 2.11: Schematic representation of demo boards power supply stage

interconnected one another, so that many different ways of sequencing the power up of the output rails are available. Of course, such complex devices need a reserved programming method: this feature is provided through the PMBUS connector, to which each device is interfaced. By connecting a PC running Fusion Digital Power Designer to the PMBUS port on the board, via its specific USB interface adapter, it is possible to configure every single parameter of the power up sequence. So after the analysis of the demo boards, the power stage for Pixel-ROD board was designed and is here reported in Figure 2.12. It is made up of three different Digital PWM System Controller (UCD9248), that are all supplied by the 12V voltage input, which is taken from Molex connector and then filtered. Each one of the UCD9248 controls four DC/DC Switching regulators, which are of three different types:

- PTD08A020W, a single output switching regulator, with a maximum supplied

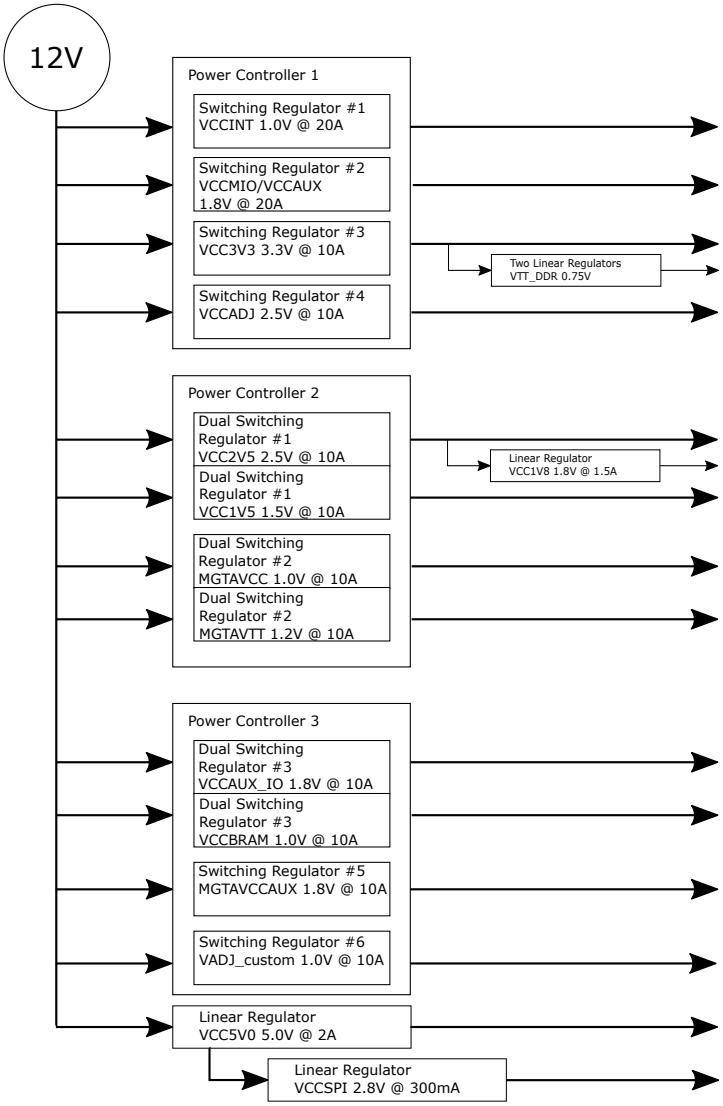


Figure 2.12: Schematic of Power Supply stage for Pixel-ROD

current of 20A;

- PTD08A010W, a single output switching regulator, with a maximum supplied current of 10A;
- PTD08D210W, a dual output switching regulator, with a maximum supplied current of 10A on each output.

Therefore, the Digital PWM System Controllers manage four rails, but only half of the

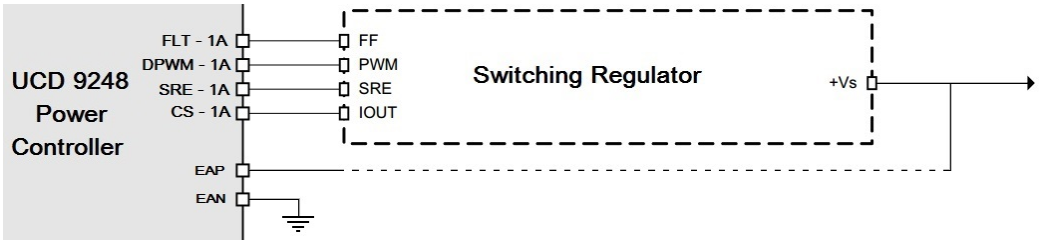


Figure 2.13: Schematic view of the interconnection between a single rail of the UCD9248 power controller and its switching regulator.

eight possible phases. In order to do so the Digital PWM System Controller needs a few signals (see Figure 2.13):

- a DPWM signal, that is the Digital Pulse Width Modulation output signal through which the switching regulator is able to define the output voltage;
- a EAP/EAN differential input signal through which the UCD9248 can read the output voltage from the DC/DC switch, therefore regulating or assessing an error whenever the read value is different from the one decided by the user;
- a CS signal, which is the Current Sense input signal, used to measure the output monitor value and eventually assess a fault if measurement is outside the user-defined range;
- a FLT signal, which is the input Fault signal used to assess an error on the actual monitored values, with respect to user-defined parameters.

2.3.1 Prototyping

From the analysis of all the required voltages and currents of the Pixel-ROD board from Figure 2.12, it is possible to determine the maximum required power from the board:

$$P = \sum_i V_i I_i = 242W$$

This means that the board has to be realized in order to withstand a maximal input current of 20 A on the 12 V Molex connector, which is a considerable value of current

for this type of applications. The solution to this level of criticality is achieved through a good *stackup*.

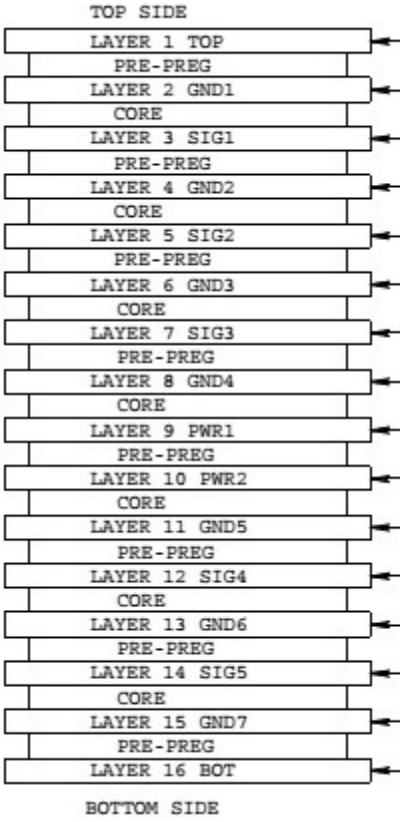


Figure 2.14: Stackup of Pixel-ROD.

The stackup defines the composition, the thickness and the function of each layer of a Printed Circuit Board (PCB). In the Pixel-ROD case a safe stackup was adopted, which is shown in the Figure 2.14: a 16 layers PCB was chosen in order to provide the necessary space to the high number of traces in the board; furthermore, the required level of insulation is achieved by alternating signal layers with ground layers, as well as concentrating the power layers into the innermost section of the board.

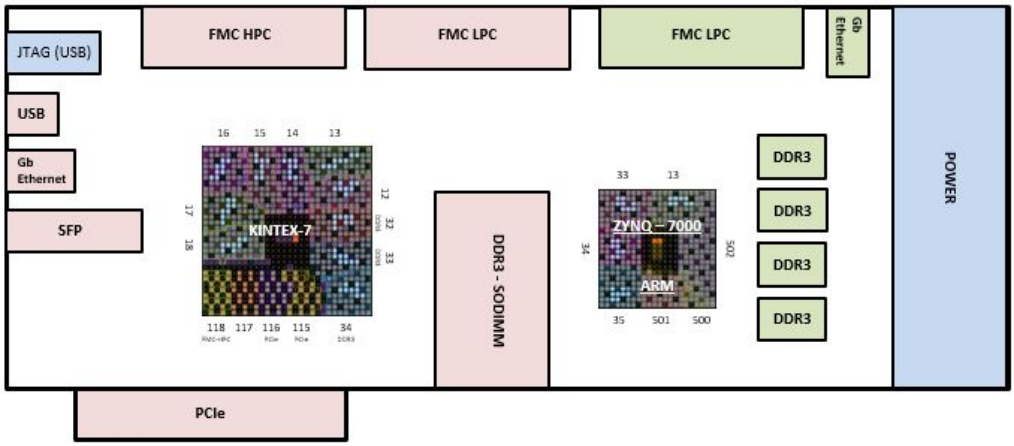


Figure 2.15: Pre-Layout of Pixel-ROD

Given all these features, a layout was proposed for the Pixel-Rod board, which is represented in Figure 2.15. This proposed layout is pretty similar to the one of KC705 demo board, but with some adjustments: firstly, the SFP connector is moved on the left edge of the board, in order to shorten the length of the traces that connect the SFP to Kintex FPGA, thus allowing the possibility to reach a nominal speed of 9.6 Gbps; the power section has to be implemented on the right side of the boar, therefore being distant from critical devices, like DDR3 SODIMM, or again the SFP connector. Before

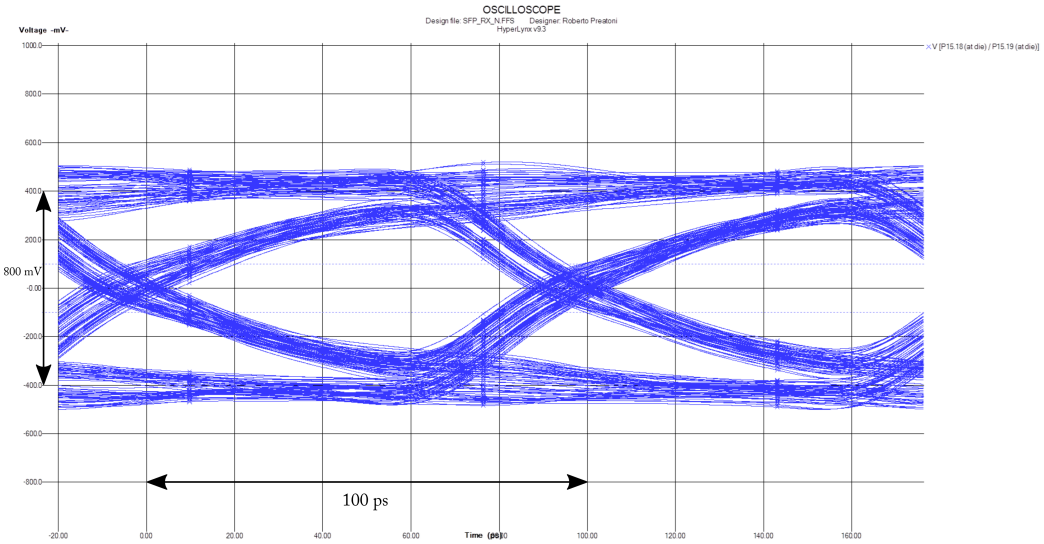


Figure 2.16: Eye diagram for SFP signals at the maximum nominal data rate (9.6 Gbps).

the fabrication of the board, a simulation was performed for the SFP connector, in order to verify the absence of major faults on this high-speed signal. By means of Hyperlinks software from Mentor Graphics, it was simulated a signal with data rate of 9.6 Gbps through the designed SFP traces and connector. The result, reported in Figure 2.16, shows what is called *eye diagram*: by simulating iteratively the signal over its traces, triggering on both rising and falling edges, it is possible to control the whether the periodicity of the signal is preserved or not. Many factors can damage the integrity of the signal, such as interferences with other signals or even noisy power supplies, making it shift or stretch between different cycles. These factors cause the shrinking of the blank area between signal's levels (named *eye*), therefore deteriorating the signal itself. In Figure 2.16 it is possible to notice that the *eye* is open, meaning that the signal is preserved, even at such high data rate.

Another constraint in the realization of the board was its size. Since the Pixel-ROD is intended to fit a PC case, as already stated above, the maximum length for the board has been set to 30 cm, thereby adding little space for the device placement. Furthermore, since the Pixel-ROD is a PCIe board, also a constraint on the thickness was added, thus fixing the maximum number of layers to 16, in order to respect the PCIe standards. Finally, the height of the board was left free, to allow sufficient room for all the devices necessary on Pixel-ROD.
The result of all these efforts are hereafter shown, in Figure 2.17.

As the figure shows, the board height has been essentially increased, providing more room for the power supply Molex, as well as for the FMC LPC connected to the Zynq. Furthermore, in order to be compliant to VITA 57.1 FMC standard, the area above the two adjacent FMC connectors has been left empty.

Two prototypes of Pixel-ROD have been produced, in order to have a faster debug and hardware *wake up* phase, which is the topic of discussion of the next chapter.

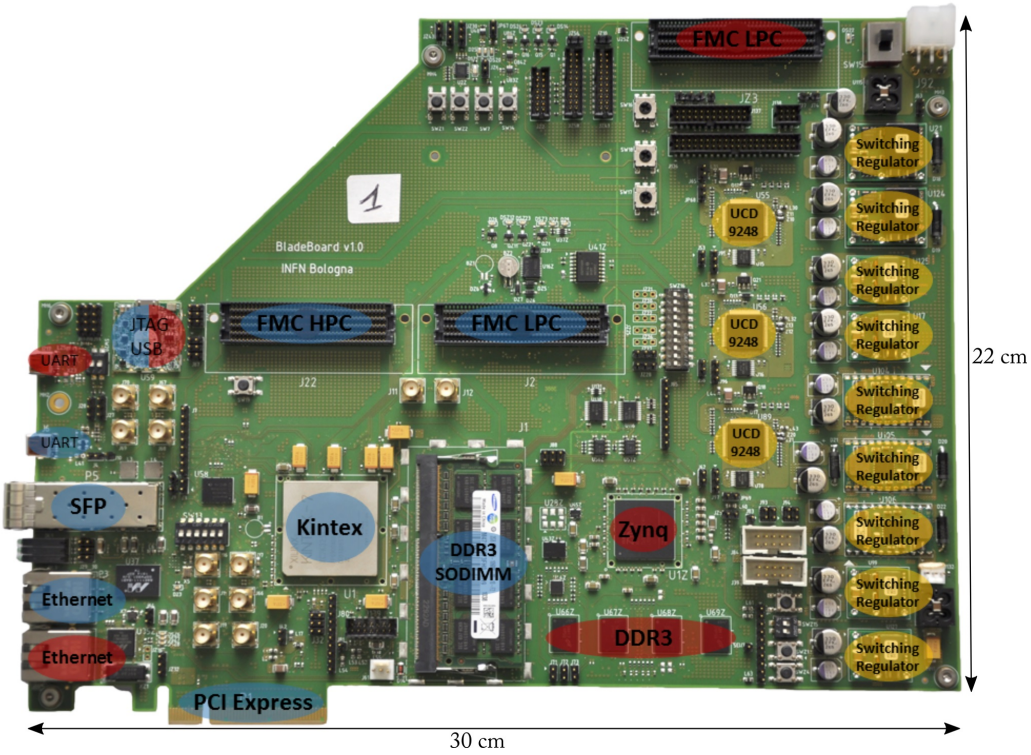


Figure 2.17: The Pixel-ROD prototype. The components highlighted in blue are those connected to Kintex FPGA; the red ones are linked to the Zynq FPGA. The components highlighted in yellow belong to the power stage.

Chapter 3

Pixel-ROD tests results

Complex electronic boards, such as the Pixel-ROD, need a careful testing and *hardware wake-up* phase, especially when just at the prototype stage. With the expression *hardware wake-up* are here intended all the actions needed to assure that, when the board is powered up for the first time, the hardware is working correctly. This first step is needed in order to prove that the hardware is capable of being correctly configured, hence providing a suitable environment where major faults of the board are excluded and the user's firmware can be actually tested.

The work that has been done on the Pixel-ROD can be mainly divided into 2 stages: the first goal has been the correct configuration of the board power up, which passes through the configuration of all three UCD9248 power controllers; the second goal has been the validation of devices and functionalities of Pixel-ROD, by implementing tests that aim to verify the expected performances. Therefore, a detailed description of the steps stated above is hereafter reported.

At present - March 2017 - all fundamental devices have been tested on both prototype boards, with the exception of FMCs and PCIe connectors, since a specific instrumentation is needed in order to verify these interfaces.

3.1 Power supply debug and test

As previously stated, when such highly complex prototypes are produced with programmable power supply devices, the act of switching on the board turns out to be

a non-trivial and high risk task. Therefore, during the design phase, the board has been given a specific feature to prevent damage to sensitive devices, like the FPGAs. In order not to spread uncontrolled voltages all over the board, the power stage was sectioned from the rest of the board by means of *solder pads*, shown in figure Figure 3.1.

Solder pads are basically interruptions between adjacent power planes that are expected to be connected during normal operations; since they bring this interruption on a reachable layer (top or bottom of the board), they allow the soldering of the disjointed planes by means of a drop of tin or a 0 Ω resistor.

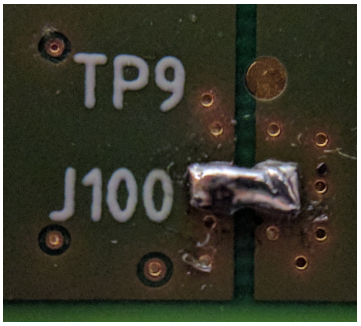


Figure 3.1: Solder pad example.

Starting with all open solder pads, we powered up the Pixel-ROD prototype number one by means of an external power supply, while monitoring it by means of an infrared camera and limiting the input current to 1 A, in order to prevent major faults. As soon as we powered up the system the generator reached the current limit, hence providing us with the information that something was wrong.

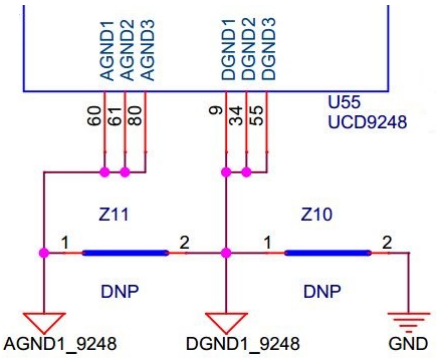


Figure 3.2: Schematic detail of mistaken connections.

The reason behind this first fault is shown in Figure 3.2: a mistake during the design phase allowed both digital and analog grounds to be left floating, due to components *Z10* and *Z11* being erroneously declared as *Do Not Place* (DNP). Luckily, this problem was quickly identified and fixed, by welding them to the general ground plane.

An almost identical problem was found on the EAN signals (see signal description in 2.3): these nets need to be referred to the ground, in order to provide a correct read value for the UCD9248. Further checks on the generated *netlist* (the file listing each existing trace on the board) proved that all of these nets were not actually tied to the common ground, but rather to another reference with the same name, that the designing software had created. As a result, this error was causing a misleading read of the voltage value by the power controllers. Again, this problem was solved by patching the board with an

external wire connected with the ground signal provided via Molex input connector.

Aside for a DC/DC switch mounted backwards and for a couple of feedback signals (EAP/EAN differential pairs) that were reversed, thereby providing feedback to the wrong rail, a couple of more sneaky issues were identified and patched. For what concerns the first one, we realized that the sectioning of power supply stage was too much aggressive: in other words, the solder pads were placed way too upstream, hence excluding not only the sensitive devices of Pixel-ROD, but also the voltage feedbacks to the UCD9248 power controllers, as shown in Figure 3.3 with reference to Figure 3.8. This caused the controllers to be unable of correctly configure the DPWM signal, which is used to control the DC/DC output voltage. The last mistake and maybe the most tough to identify,

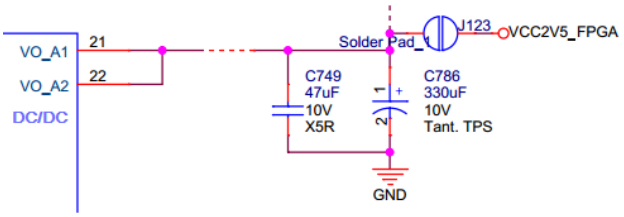


Figure 3.3: Schematic detail of the overly aggressive power sectioning. The solder pad $J123$ placed before the power output, that excludes voltage feedback too.

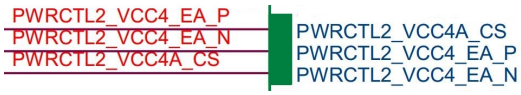


Figure 3.4: Schematic detail of last mistake identified.

is shown in Figure 3.4: on the left side are reported 3 signals coming from one of the three UCD9248 controllers; on the right side the same three signals from the DC/DC switching stage. It is possible to notice that these connection are switched one another, therefore implementing a mistaken connection from the DC/DC switching stage to the UCD9248. It was particularly hard to identify this issue because of the number of nets involved: from each power controller 80 nets go to the DC/DC switching stage. Furthermore, in the schematic design of the Pixel-ROD each power controller is described but not straightly connected to its DC/DC switches. This is due to the fact that in the making of the schematic design for these highly populated boards, a *hierarchical* structure is used, where each device is described in a single page. In this type of schematic, a specific page usually named *Top Module* or just *Top*, is reserved to trace and sum up the connections between devices described in their specific page. The mistake shown in

Figure 3.4 had to be found in the Top Module of Pixel-ROD schematic design, among all the connection between the high number of devices that make up the board.

After this long debug period, we could finally program the three UCD9248 power controllers by means of *Texas Instruments Fusion Digital Power Designer*.

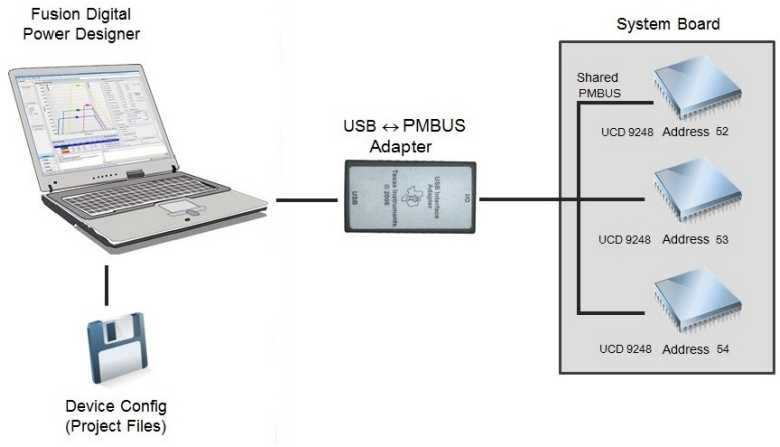


Figure 3.5: Schematic representation of power controllers programming setup.

A brief representation of the set-up needed to configure these devices is reported in Figure 3.5. Firstly, a Universal Serial Bus (USB) to Power Management BUS (PMBUS) adapter is used to interface the PC with the Pixel-ROD's PMBUS reserved header connector. The PMBUS connection is an open standard protocol that defines a bus for communicating with power conversion and other devices. It is based on the System Management Bus (SMBUS), which is a single-ended two-wire bus, derived from I^2C for the purpose of lightweight communication. As shown in Figure 3.5, each device present on the Pixel-ROD board has its specific address, which is hardware-coded by means of two resistors.

At power-up, the power controller applies a bias current ($I_{BIAS} = 10\mu A$) to each address detect pin,

PMBUS Address	$R_{PMBUS}(k\Omega)$
11	205
10	178
9	154
8	133
7	115
6	100
5	86.6
4	75
3	64.9
2	56.2
1	48.7
0	42.2

Table 3.1: Address resistor code.

and the voltage on that pin is captured by the internal 12-bit ADC. The equation that gives the selected address is the following:

$PMBUSAddress = 12 \times PMBUSAddress1 + PMBUSAddress0$

In Table 3.1 it is shown the equivalence between the resistor placed on the board and the associated address. As Figure 3.5 reports, on the Pixel-ROD board the UCD9248 devices have the following addresses, respectively: 52 ($R_{PMBUS_1} = 75k\Omega$, $R_{PMBUS_0} = 75k\Omega$), 53 ($R_{PMBUS_1} = 75k\Omega$, $R_{PMBUS_0} = 86.6k\Omega$), 54 ($R_{PMBUS_1} = 75k\Omega$, $R_{PMBUS_0} = 100k\Omega$).

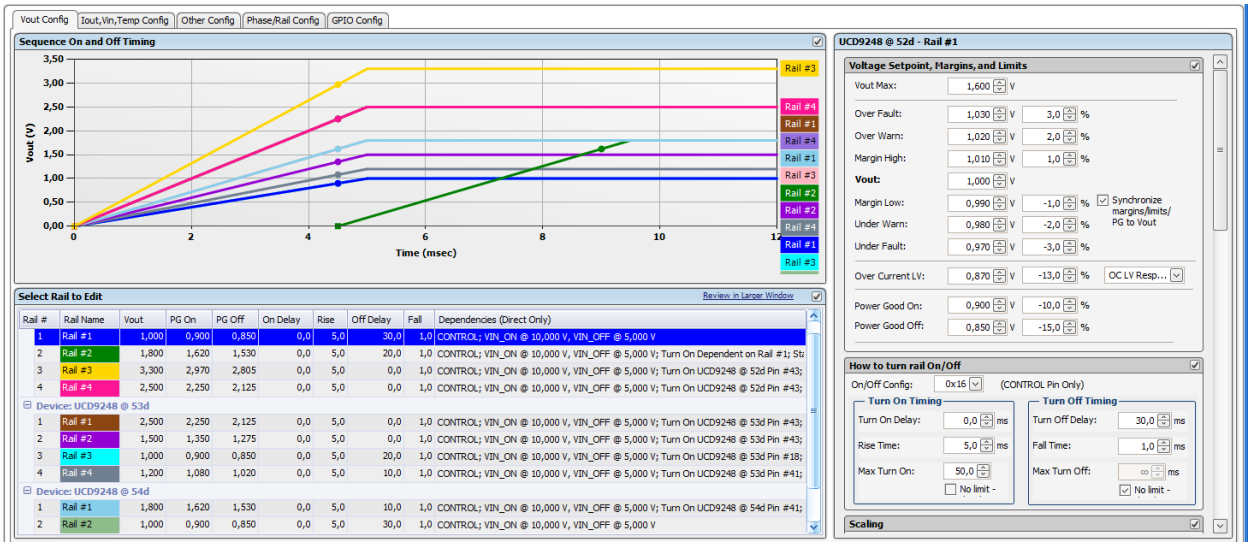


Figure 3.6: Starting page of the Fusion Digital Power Designer GUI, showing configuration parameters for every lane.

Once the set-up described above has been prepared, the system is ready to be configured with a user-defined new project. In order to do so, the GUI requires the user to enter some parameters (see Figure 3.6) relative to the designed system [18], of which the most relevant ones are hereafter described:

V_{OUT} It is the desired voltage for the selected rail, as well as its defining parameter. It is expressed in Volts.

V_{OUT} Over/Under Voltage Fault These two values define what the controller sees as a voltage fault value on the line, thereby causing it to take the user-defined action in case of asserted fault. The specific range for each output voltage rail is obtained from the datasheets of the devices that are supplied by the rail; whenever

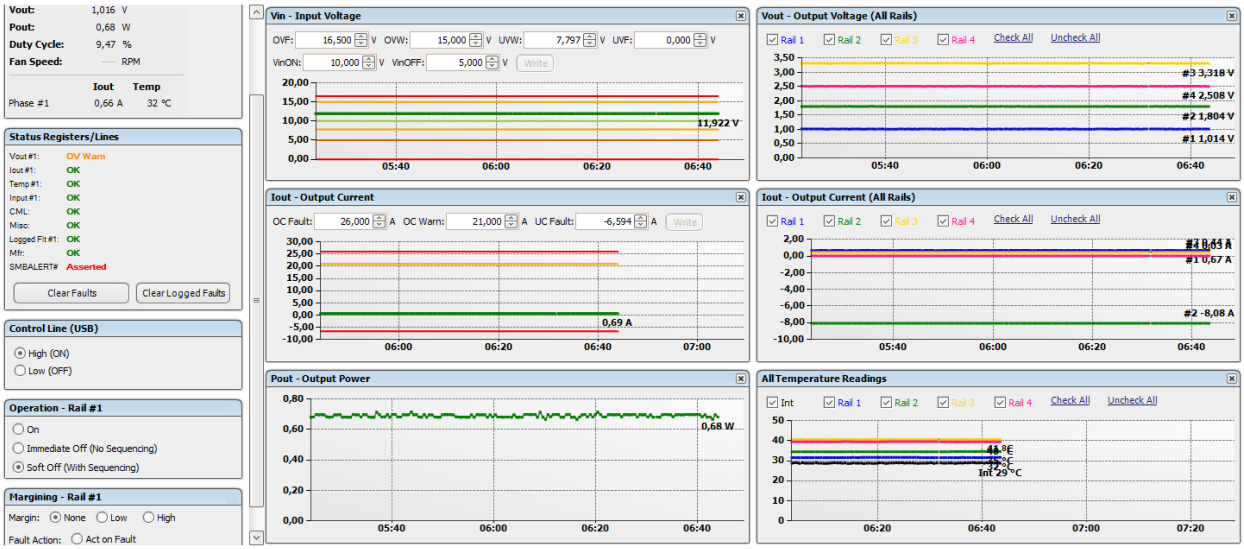


Figure 3.7: Monitor page of the Fusion Digital Power Designer GUI. This page allows to check the sensed voltages, currents and temperatures of each attached power controller. Furthermore, it shows the status of the register, where the possible faults are reported and differentiated.

different ranges are available, the stricter one is applied. These parameters can be expressed in Volts as well as percentage gap from V_{OUT}.

PGood On/Off The Power Good (PGood) On sets the voltage at which the controller can assess that the selected rail is working; conversely, PGood Off parameter is used to set the voltage at which the UCD9248 reads the rail as not working. Whenever the read voltage for each rail is above its PGood On value, a global PGood active-low signal is assessed on the PMBUS by the controller. These parameters can be expressed in Volts as well as percentage gap from V_{OUT}.

On/Off Configuration This parameter sets the activation mechanism the rail output. Four different set-up are available:

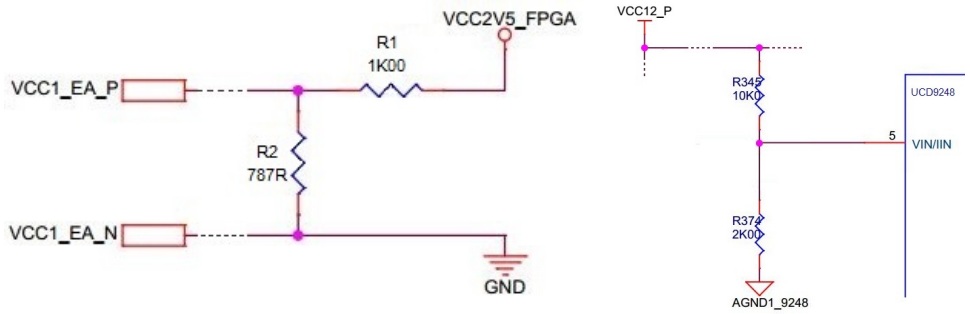
- **Always converting**: As soon as the controller is powered up, it activates the rail;
- **Operation pin**: the controller turns up the rail when the operation command is assessed from the GUI.

- **Control pin**: the controller turns up the rail when the operation command is assessed from the GUI. Differently from the previous option, this one depends on the presence or absence of the USB to PMBUS Adapter, since if it is present the Control pin is pulled low, otherwise it is pulled up.
- **Both Control & Operation pin**: the controller turns up the rail whenever both Control and Operation pins are activated from the GUI.

Whenever operating with Control or Operation pins it is also possible to decide the polarity of the signal, defining whether it is active-high or active-low.

T_{ON} Delay This parameter defines the time delay after the *On/Off Configuration* signal, after which the rail is activated. It is expressed in milliseconds.

Rise Time This parameter defines the time within which the rail rises from 10% to 90% of V_{Out}. It is expressed in milliseconds.



(a) *Voltage divider for V_{OUT} Scale Loop*

(b) *Voltage divider for V_{IN} Scale*

Figure 3.8: Voltage divider used on UCD9248 power controllers.

V_{OUT} Scale Loop This parameter is very important since it controls how the controller compares the sensed voltage to the actual parameter set in V_{OUT}. This parameter is the *transfer function* of the voltage divider (see Figure 3.8a) that is present on each EAP/EAN differential signal of the controller, which is given by the function: $V_{OUT}ScaleLoop = \frac{R_2}{R_1 + R_2}$. When checking the voltage level the controller scales both the read voltage value and the commanded V_{OUT} value by V_{OUT} Scale Loop;

then these two scaled values are passed to an *Error Processing and Control Loop* that compares them and assesses whether they are the same or not.

V_{OUT} Scale Monitor This parameter works just as the one reported above, but is used to correctly scale the monitored voltage (see Figure 3.7), in order to provide the right representation of it. V_{OUT} Scale Monitor depends on the voltage divider implemented on the rail too, therefore it always has the same value of V_{OUT} Scale Loop.

I_{OUT} Cal Gain & Offset These two parameters are used to set the calibration and the offset of the sensed current. They can be both obtained automatically by Fusion Digital Power Designer.

Temp Cal Gain & Offset These two parameters are used to set the calibration and the offset of the sensed temperature.

Over Temperature (OT) Fault This parameter defines the temperature threshold above which the controller assesses a fault and stops the operations. It is expressed in °C.

I_{OUT} Over Current & Under Current Fault These parameters define the current values at which the controller issues a fault. They can be expressed in Ampere, or as percentage of a *rated* current, or automatically assigned by the GUI once the DC/DC switching associated is defined.

V_{IN} Over/Under Voltage Fault As the names suggest, these two values are used to define the range of the input voltage V_{IN}. An input voltage outside this range causes the controller to stop operation and assesses the related fault (Over/Under Voltage). Both values must be expressed in Volt.

V_{IN} This parameter simply defines the value of the input voltage to the controllers. It is expressed in Volt.

V_{IN} Scale As already described for the V_{OUT}, also the V_{IN} is scaled before being read by the controller. This parameter defines the scale factor that is implemented in hardware by means of a voltage divider, which is placed before the controller V_{IN}

input pin. With reference to Figure 3.8b, this parameter can be calculated with the following equation:

$$V_{IN}Scale = \frac{R_{374}}{R_{374} + R_{345}} = \frac{2 \cdot 10^3\Omega}{10 \cdot 10^3\Omega + 2 \cdot 10^3\Omega} \simeq 0.167$$

T_{ON} & T_{OFF} Controller These two parameters define the voltage threshold at which the controller is Turned On (T_{ON}) or Turned Off (T_{OFF}). Both values are expressed in Volt.

Even though these controllers proved rather complicated to configure, their role is fundamental for the correct functionality of the board. The new and complex FPGAs implemented on Pixel-ROD board require many different supply voltages, that need to be stable but especially included in a very tight range. For example, the V_{OUT} Over/Under Voltage Fault parameters for VCC_INT, the core voltage for both FPGAs, it is set to a value of 1.0 V and constrained into a range of ±3% from it.

Nevertheless, the most important feature that these controllers add to the power stage of Pixel-ROD board is the ability to set a user-defined sequence of power up for each voltage rail. Since the controllers are interconnected, it is possible not only to sequence the power up within the same controller, but also between different controllers as well. As Figure 3.9 shows, the GUI features a section where it is possible to select

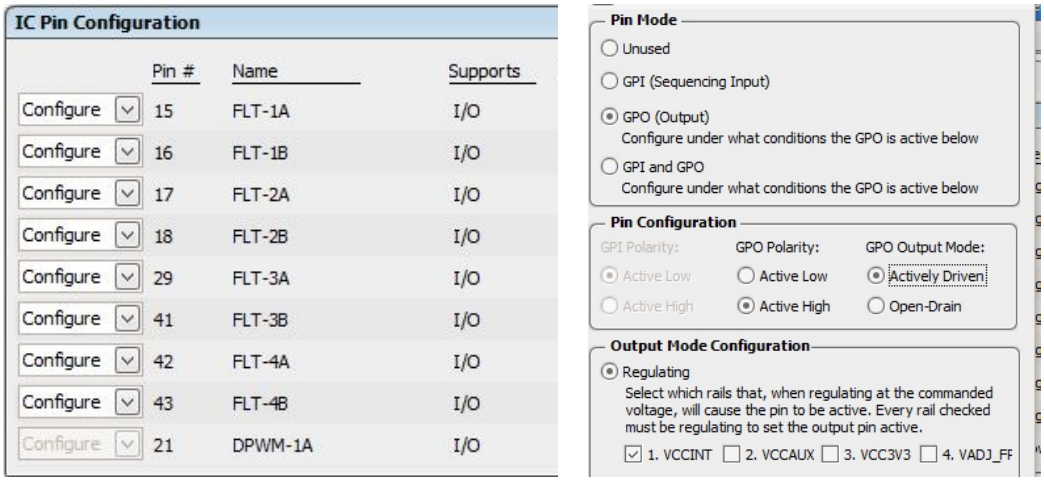


Figure 3.9: GUI interface page for sequencing options.

a user-defined action that a configurable pin shall undertake. Not all the controllers' pins are available, since a good portion of them is used for the control of rails. Once a free pin is identified, it is possible to configure it by means of the window shown on the right side of Figure 3.9: firstly, a *Pin Mode* has to be defined, by declaring if whether this pin is either an input or an output; secondly, polarity and output mode (available only if the pin is actually an output) have to be selected, thereby defining if the pin is active-high or active-low and actively driven or open-drain; finally, it is possible to select when this signal shall turn active. The example reported in Figure 3.9 shows a signal that is active when the *VCCINT* rail rises above 90% of its output voltage value. For

Controller Address	Rail n°	Rail Name	Voltage Value	Sequence Order
52	1	VCCINT	1.0V	1
	2	VCCAUX	1.8V	2
	3	VCC3V3	3.3V	4
	4	VADJ	2.5V	4
53	1	VCC2V5	2.5V	4
	2	1V5VCC	1.5V	4
	3	MGTAVCC	1.0V	2
	4	MGTAVTT	1.2V	3
54	1	VCCAUX_IO	1.8V	3
	2	VCCBRAM	1.0V	1
	3	MGTAVCCAUX	1.8V	–
	4	VADJ_CUSTOM	2.5V	–

Table 3.2: Voltage Rail parameters and sequence

Pixel-ROD board the power up sequence is made up of four different stages, as reported in Table 3.2; some rails do not need to be included, therefore their sequence order is omitted. It is important to notice that it has been defined a power off sequence as well, which follows exactly the reversed order stated in Table 3.2.

The final result of such configuration was controlled with an oscilloscope, in order to verify that the expected sequence was actually being implemented by the power controllers. The power up sequence observed with the oscilloscope is reported in Figure 3.10: a voltage rail from each step of the starting sequence has been included and is possible to note that it works as expected, since every rails (with the exception of the first) starts when the previous one is correctly turned on.

Finally, after completely defining the power up sequence, the Pixel-ROD boards was

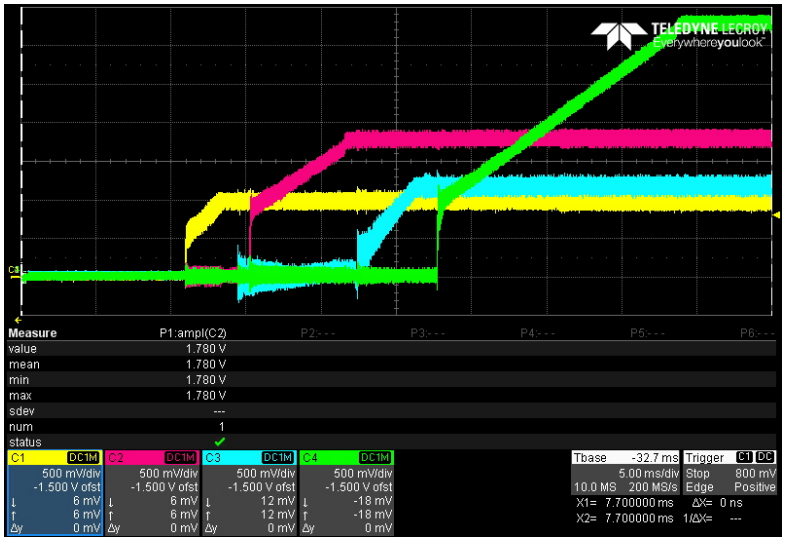


Figure 3.10: Power on sequence observed with the oscilloscope. The yellow line corresponds to *VCCINT*, the pink one to *VCCAUX*, the cyan one to *MGTAVTT* and the green one to *VCC3V3*.

successfully turned on and the configuration files for the UCD9248 were stored in the built-in flash memory of the controllers, thereby removing the need to use the GUI in order to do so.

3.2 Kintex-Zynq internal bus test

After having consolidated the functionality of the power stage of Pixel-ROD, a first test had to be designed, in order to prove the possibility to perform some basic functions on the board. Therefore, it was decided that this test had to prove the correct programmability, via JTAG, of both the FPGAs (on the Zynq side, only the PL has been used for the test).

To perform this test, as well as all the others reported in this thesis, we took advantage of the new *Vivado Design Suite*, which is the software tool produced by Xilinx for synthesis and implementation of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. In order to achieve the proposed task it was decided to implement a test using the

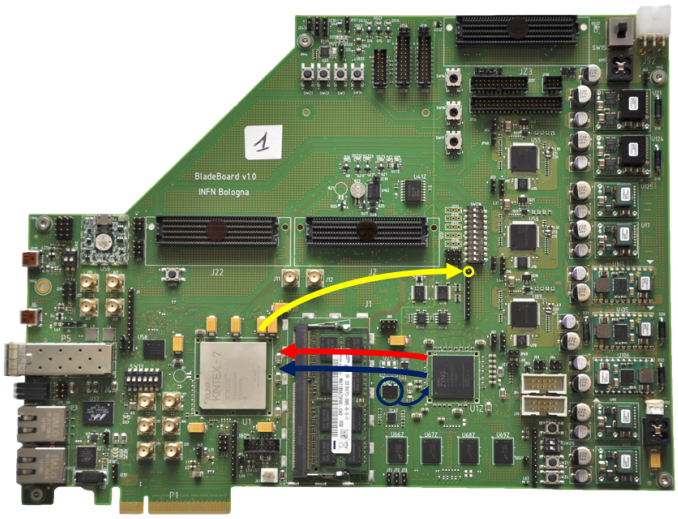


Figure 3.11: Schematic description of the test's behaviour. The blue arrows refer to clocking path, with clock source circled in blue; the red arrow refers to counter's path; finally, the yellow arrow indicates the path of the control signal.

differential bus between the FPGAs. The test is schematically represented in Figure 3.11: the 200 MHz clock IC (circled in blue) is used as clock source for the Zynq PL, where a simple counter is implemented and sent to the Kintex-7 through the whole 21-bit-wide differential bus (namely, *KZ-Bus*); furthermore, the clock generated by the external IC is passed to the Kintex-7 through a reserved differential lane. The Kintex-7 receives both the clock and the counter and implements a simple logic: it takes the value provided on the KZ-Bus by the Zynq at a certain clock cycle and subtract it to value read on the same bus at the successive clock cycle: if the difference between these two values is a logical "1", then the Kintex sends a logical "0" over a GPIO header (circled in yellow in Figure 3.11). When all the bits in the KZ-bus are set to "1", the Zynq FPGA resets the count and starts again from a logical "0": in this way, each time the Zynq finishes to count, the subtraction performed by the Kintex will not be a logical "1", thus forcing the signal on the GPIO header to a logical "1" as well. Since in the test a 200 MHz clock source has been used, we expected the signal on the GPIO header to rise accordingly to:

$$t_{period} = 2^{21} \cdot \frac{1}{200 \cdot 10^6 Hz} \simeq 10.5ms$$

The simulation shown in Figure 3.12, represent the result of the firmware produced for

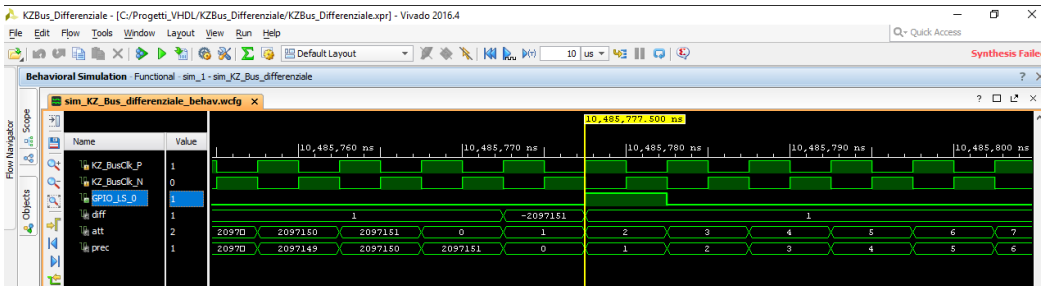


Figure 3.12: Result of test's simulation for Kintex FPGA using Vivado software.

the Kintex in this test: the first two lines represent the clock that the Zynq sends to the Kintex (blue arrows in Figure 3.11); the third line is the GPIO signal (yellow arrow in Figure 3.11) described above. Lastly, the remaining signals, from bottom to top, are: the last(*prec*) value and the current (*att*) value read on the KZ-bus and the difference (*diff*) between them. As one can notice, the global behaviour of the test is just as reported above, thereby allowing us to configure both FPGAs with the firmware.
By means of Vivado we loaded the configuration files on the FPGAs and connected the oscilloscope to the GPIO header, in order to check the presence of the expected signal. The Figure 3.13 shows exactly what was expected: by means of the oscilloscope we observed on the GPIO header a spiking signal, thereby assessing the successful validation of the internal bus at 200 MHz frequency.

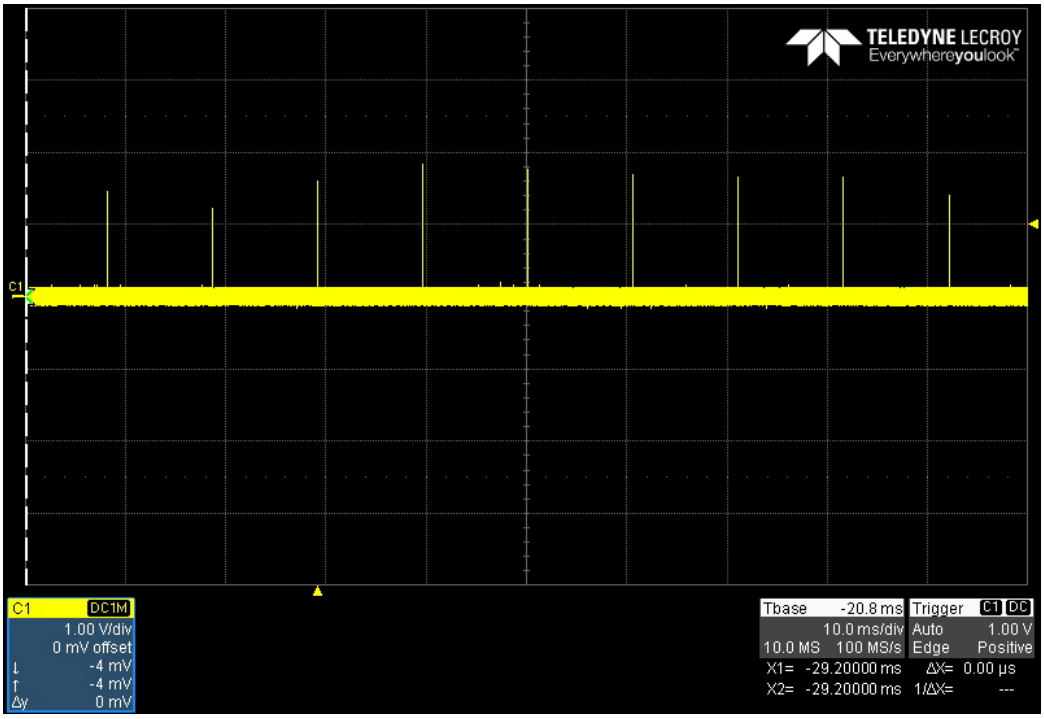


Figure 3.13: Result of KZ-bus test obtained with the oscilloscope in GPIO header.

3.3 Kintex interfaces and memory test

During the validation of such complex board, the debug and test phase of memories and attached interfaces is always a decisive step, since it brings the board closer to implement its full functionality. However, this kind of tests that involve not only the bare trace interconnection between devices, but also specific ICs and especially the development of more complex firmware, often result to be very time consuming. In this way, the smart design obtained taking KC705 board as reference (see 2.3), speeded up the whole process, since it made available a platform identical to the Pixel-ROD board, where the firmware could be validated before being loaded on the tested board itself.

3.3.1 Vivado IP Integrator and AXI4 Interface

To develop the firmware of this test the *Intellectual Property (IP) Integrator* tool from the Vivado Design Suite has been used. This tool lets the user create complex system designs by instantiating and interconnecting *IP cores* from the *Vivado IP catalog* onto

a design canvas. In this way the user can take advantage of the IP already available in the Vivado library to speed up the firmware development, which otherwise would take a consistent amount of time. Therefore, before going into further details of this test, it is necessary to provide a brief description of the main IP cores used and the AXI interface.

The AXI protocol

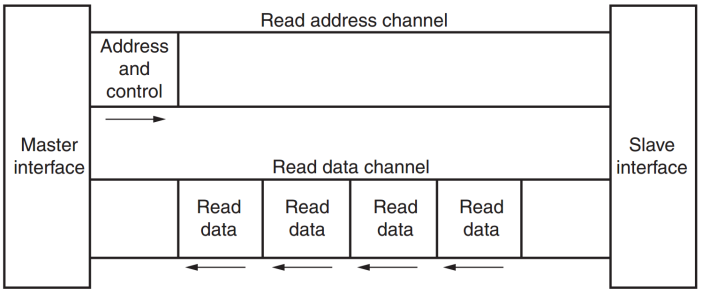
The Advanced eXtensible Interface (AXI) protocol [19] is part of the ARM Advanced Microcontroller Bus Architecture (AMBA), a family of micro controller buses first introduced in 1996. The first version of AXI was included in AMBA 3.0, released in 2003; the second version of AXI, the AXI4, is included in AMBA 4.0, released in 2010. The AXI4 protocol is the default interface for IP cores, extensively used during the debug of the Pixel-ROD board. The AXI4 protocol presents three main features, making it a good choice: firstly, it provides a *standardized interface* between many IPs, therefore allowing the user to concentrate on the system debug rather than the protocol needed; secondly, the AXI4 protocol is *flexible*, meaning that it suits a variety of applications, from single, light data transaction to bursts of 256 data transfers with just a single address phase; finally, since the AXI4 is an industrial standard, it allows the access also to whole ARM environment.

There are three types of AXI4 interfaces:

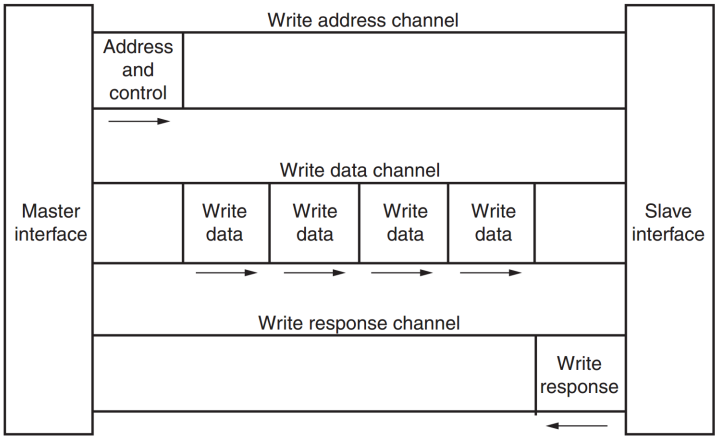
- AXI4, used for high-performance memory-mapped operations;
- AXI4-Lite, used for simple, low-throughput memory-mapped communication;
- AXI4-Stream, used for high-speed data streams.

The AXI4 interface uses a *Master-Slave* architecture, thereby instantiating a communication where the master device sends commands or data to a slave device. Furthermore, all AXI4 masters and slaves can be connected together by means of a specific IP, named *Interconnect*. Both AXI4 and AXI4-Lite interfaces consist of five different channels:

- Read Address Channel;
- Read Data Channel;
- Write Address Channel;



(a) *Channel architecture of AXI read operation.*



(b) *Channel architecture of AXI write transaction.*

Figure 3.14: Architecture of AXI transactions.

- Write Data Channel;
- Write Response Channel.

Data can move in both directions between master and slave simultaneously and data transfer sizes can vary. The limit in AXI4 is a burst transaction of up to 256 data transfers, while the AXI4-Lite interface allows only 1 data transfer per transaction. As shown in Figure 3.14, both the write and read transactions have a very simple architecture. When the master needs to read data from a slave, it sends over the dedicated channel both the address that needs to be read and the control command; the slave answers by sending over the dedicated channel the result of the read operation. Indeed, when the master needs to write into a specific register or address of a slave device, it sends again over the dedicated channel both the address to which the write operation has to

be performed, as well as the control command. The master passes over the Write Data Channel all the informations that need to be written and the slaves answers, on the Write Response Channel, assessing whether the operation was successful or not.

The AXI4-Stream interface is different from both the previous two: it only supports a single channel (Write Data Channel) that can burst an unlimited amount of data.

Microblaze

The *Microblaze* IP core [20] is used to implement the 32-bit soft processor, indeed named Microblaze, whose symbol is represented in Figure 3.15. On the left side of Figure 3.15

Figure 3.15: Symbol of the Microblaze soft processor.

a few input signals are reported:

Clk and Reset These inputs are the reference clock and reset signals used among the whole design. The maximum clock frequency for a Microblaze implemented on a Kintex-7 is 393 MHz.

Interrupt This is the interrupt port to which every other capable IP core sends its interrupt signal. This signals is used to inform the Microblaze of the need to undertake an action to respond to the event that caused the interrupt. For example, when the Microblaze instructs another IP core to perform a *write transaction*, the end of such operation generates an interrupt signal from the IP core to the Microblaze, which is managed by the soft processor according to the logic defined by the user.

Debug This input is always interfaced with another IP core, named Microprocessor Debug Module (MDM), and is used to interface the soft processor with the *Xilinx System DeBugger* (XSDB), through the JTAG port of the FPGA for debugging purposes. Thanks to XSDB it is possible to send to the Microblaze specific instructions, that can be very useful during the debug phase: for example, the *mrd* command instructs the Microblaze to read what is contained at the desired memory or register address.

On the right side of Figure 3.15 some output signals are visible:

DLMB and ILMB Literally, Data Local Memory Bus interface and Instructions Local Memory Bus interface. These are two synchronous buses used to primarily access to on-chip block RAM. It uses a minimum number of control signals and a simple protocol to ensure that local block RAM are accessed in a single clock cycle.

M_AXI_DC and M_AXI_IC These two signals (Data Cache and Instruction Cache) use AXI4 protocol to interface the Microblaze processor with cache memory.

M_AXI_DP This signal (Data Peripheral) uses AXI4 interface to send data to the peripherals connected to the Microblaze, like the Ethernet controller.

AXI 1G/2.5G Ethernet Subsystem

The *AXI 1G/2.5G Ethernet Subsystem* IP core [21] is used to provide additional functionality and ease of use related to Ethernet. Based on the configuration, this subsystem creates interface ports, instantiates required infrastructure cores, and also connects these cores. The subsystem provides an AXI4 bus interface for a simple connection to the processor core, in order to allow access to the registers. Furthermore, 32-bit AXI4 buses are provided in order to transmit and receive data to and from the subsystem. Finally, the physical side of the subsystem is connected to the one on the board, which supports different interfaces, such as the Gigabit Media Independent Interface (GMII), that can provide support for Ethernet operations at 10 Mbps, 100 Mbps or 1 Gbps. To connect this IP core it is necessary to manage few inputs and outputs, that are shown in Figure 3.16 and described below:

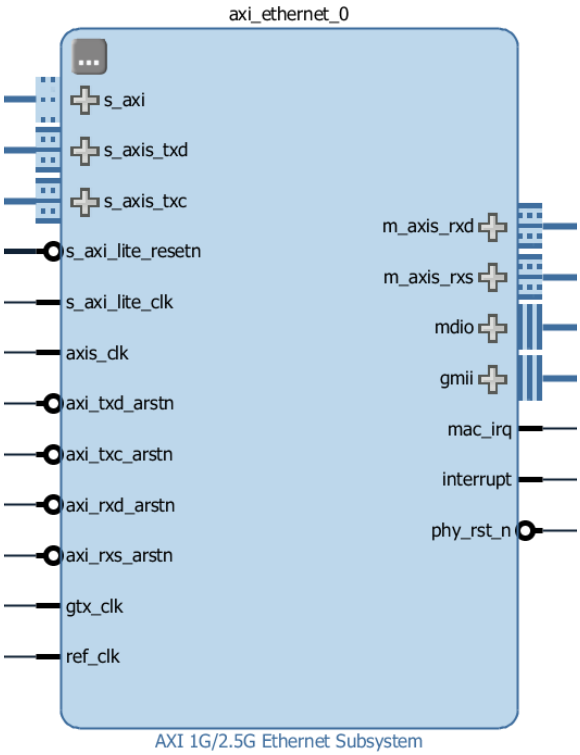


Figure 3.16: Symbol of the Ethernet Subsystem IP core.

S_AXI This AXI4 input interface is used to configure the subsystem, by accessing its registers and configuring them with user-defined options. It is paired with its clock (**s_axi_lite_clk**) and reset (**s_axi_lite_resetn**) signals.

S_AXI_TXC and S_AXI_TXD These are the reserved AXI4 input interfaces used to transmit control commands and data to the subsystem. These inputs are paired with their clock (**axis_clk**) and their reset signals (**axi_str_txd_aresetn** and **axi_str_txc_aresetn**)

M_AXIS_RXD and M_AXIS_RXS These are the reserved AXI4 interfaces used to receive data and status of the subsystem.

MDIO This is the Management Data Input/Output (MDIO) interface used to configure the PHY.

GMII This is the Gigabit Media Independent Interface, which is connected to the

board's Ethernet port.

Memory Interface Generator

The *Memory Interface Generator* (MIG) IP core [22] is a controller and physical layer for interfacing *7-series* FPGA, as well as other AXI4 slave devices, to DDR3 memory. Given the wide variety of DDR3 modules and components available, this IP core is very flexible and configurable. Aside from few unused or common signals (such as resets),

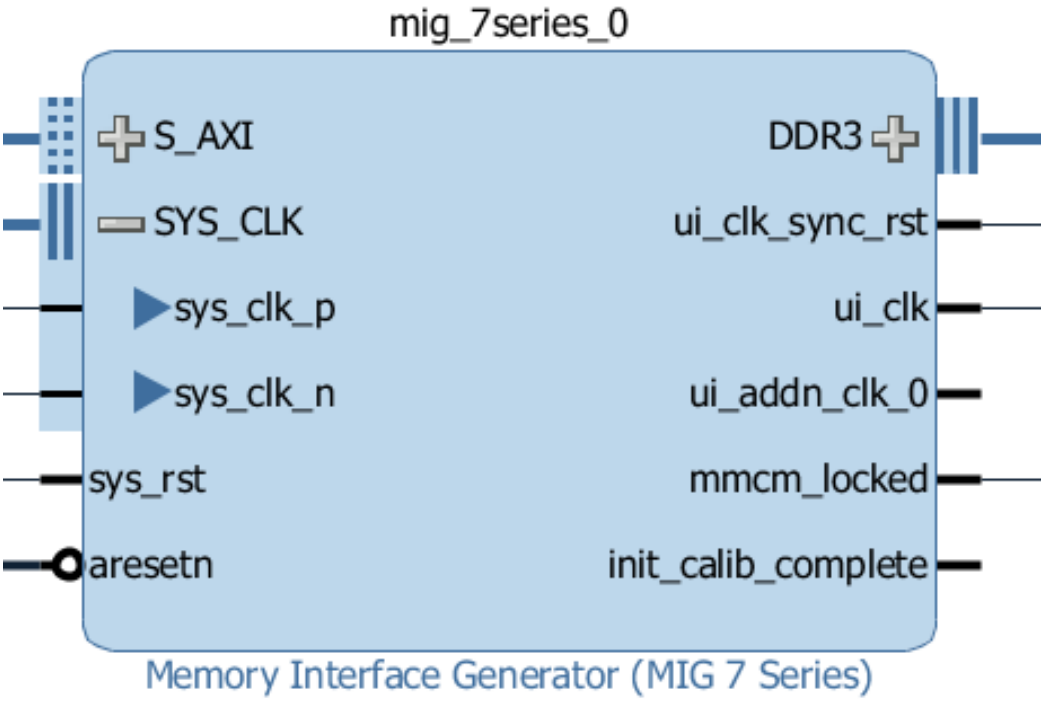


Figure 3.17: Symbol of the MIG core used in the design. The UART port on the right shows Tx and Rx signals as well, whose connection wire is not displayed since it is included in the UART port.

the IP employed used in this test and shown in Figure 3.17, has few inputs and outputs, which are described below:

S_AXI This is the input AXI4 interface which is used to interface all the transactions, such as core configuration's commands as well as the actual data that will be written on the DDR3 memory.

SYS_ CLK This is the 100 MHz or 200 MHz clock controlling the Kintex-based core (the actual possible frequency values depend on the type of FPGA). This clock is especially important because it is used to generate the reference clock for read/write operations on the memory.

ui_ clk This is the user interface output clock. It is used as reference clock all over the design and is obtained within the core by multiplying the *SYS_ CLK*. It must be either a half or a quarter of the clock frequency used to interface the DDR3 memory.

DDR3 This is the actual interface towards the DDR3 memory, supporting all data transactions. It can be configured to be a 32-bit-wide or 64-bit-wide bus.

UART

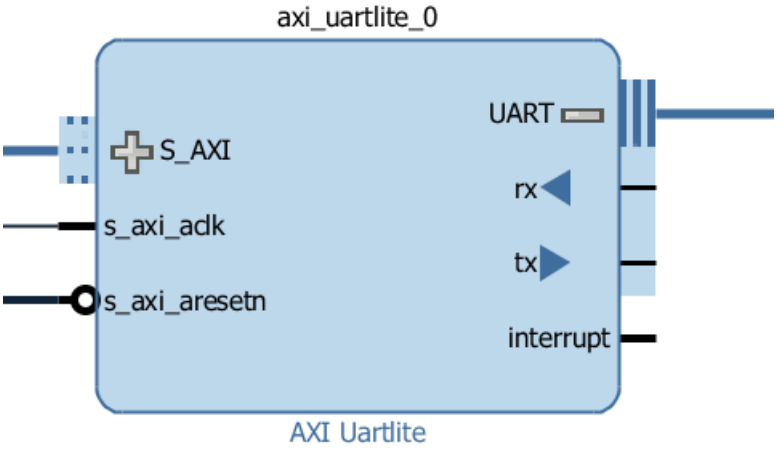


Figure 3.18: Schematic representation of UART IP core.

The *AXI UART Lite* IP core [23] is a simple and useful block used to implement the communication of the board via USB port. Excluding the common clock and reset of the AXI4 interface, it features just the two I/O ports (*S_ AXI and UART*) needed to implement the transmission and reception from and to the Microblaze.

3.3.2 Architecture of the test

The task of the test described in this section is to validate and, if needed, debug all the available interfaces. At present - March 2017 - the instrumentation available in the laboratory allows to test almost everything, with the exception of PCIe interface and FMC connectors, for which dedicated instruments are needed. In order to perform this test in the safest way, firmware and software described below have been tested before on the KC705 demo board, where every interface was correctly working, and then ported on the Pixel-ROD board. Once again, this was possible because of the smart design of Pixel-ROD: in this way, the procedure narrows down the possible mistakes to be made at each step, resulting in a more efficient and fast debug.

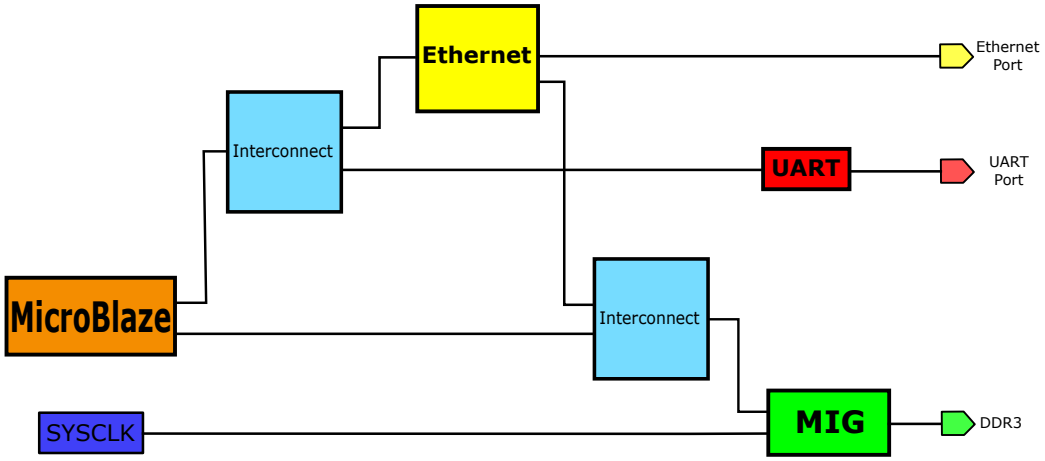


Figure 3.19: Schematic block design of the firmware implemented by means of Vivado IP cores. All connections between IP core blocks use the AXI4 interface.

The Figure 3.19 shows the schematic block design of the firmware implemented on the Kintex FPGA. The Microblaze soft processor is connected to the three interfaces under test: Ethernet, UART and DDR3 memory; the whole system works with a 100 MHz clock (referred as *SYSCLK* in Figure 3.19). The analysis of Figure 3.19 shows a good implementation of the AXI4 protocol to interface IP cores: the Microblaze, working as master, is connected to every single interface under test through *interconnect* IP cores; however, the Ethernet IP core works both as slave, receiving instructions from the Microblaze, and as master, sending data to the DDR3 module, through the second *interconnect* IP. The whole generated firmware uses a low percentage of the available

resources (see Figure 3.20), even though it is implementing already a good amount of interfaces.

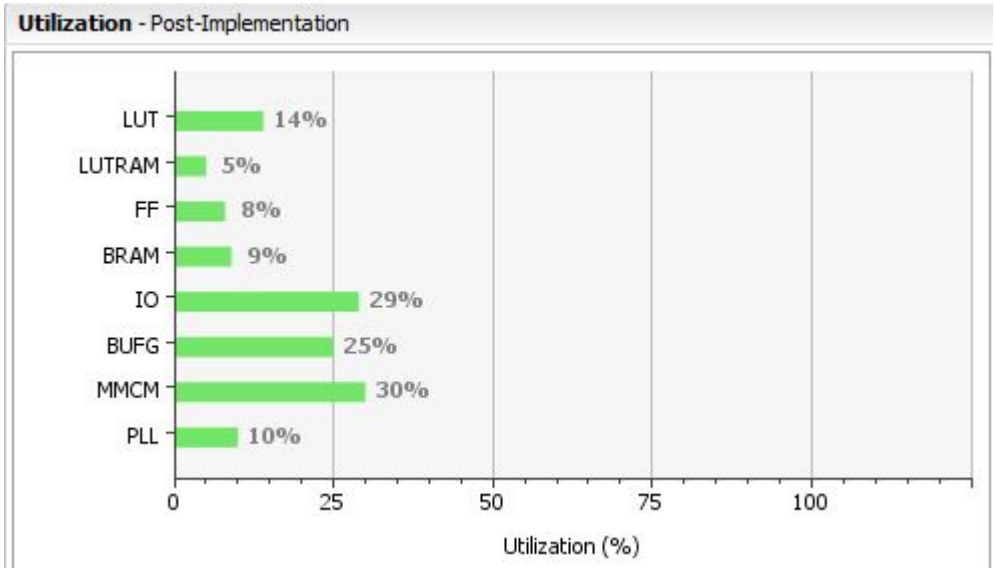
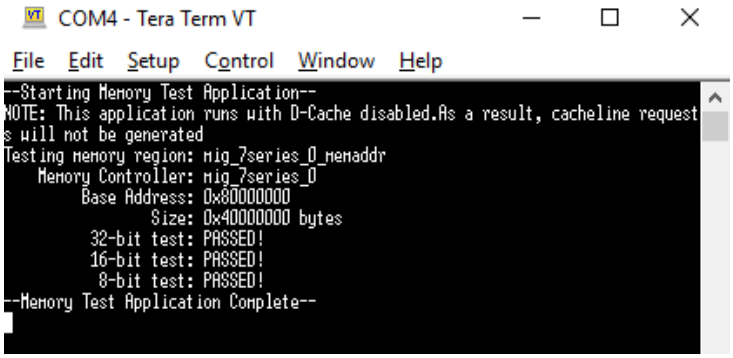


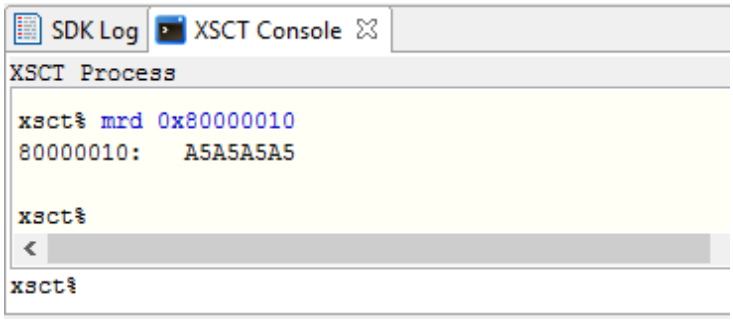
Figure 3.20: Resource utilization for the test of interfaces and memory

Nevertheless, this firmware would be completely useless without implementing the software on the Microblaze. The Vivado Design Suite offers a dedicated environment, named *Software Development Kit* (SDK), where it is possible to produce and run applications on the available board's processors. For this test two SDK built-in applications have been used: the first one has the task to test and validate the DDR3 module (thereby named *Memory Test*); the other one aims to verify the functionality of the Ethernet port. These two built-in applications allowed to immediately test the firmware developed for Pixel-ROD on KC705. Every interface works as expected. The memory test application developed for this test has a simple logic: once it has found every available memory module (along with its address), it starts writing on it a 32-bit-wide constant pattern (the hexadecimal value $AAAA5555$) over every possible memory address; if all the write transactions are performed correctly it states the success of the test through the terminal, available via UART connection. The application repeats this operations again with a 16-bit wide and a 8-bit wide constant pattern ($AA55$ and $A5$).

In order to check the success of the test, a read operation needed to be performed by the Microblaze onto the DDR3 module: by means of Xilinx Software Command-line



(a) *UART terminal response during memory test application run.*



(b) *Result of memory write control operation.*

Figure 3.21: Memory test results.

Tool (XSCT) terminal, a *mrd* command followed by the desired address was given to the Microblaze (see Figure 3.21b). The soft processor responded writing on the XSCT terminal the value read at the indicated address, thus confirming the success of the test, as already stated by UART terminal (see Figure 3.21a).

The Ethernet test application works similarly to the memory one just presented. This software implements an *echo server* on the Microblaze itself, which takes keyboard's inputs from a terminal connected to it; after that, it answers back sending to the same terminal a message identical to the inputs received. This time, the implementation of the software on the Pixel-ROD did not work at first; thus, since it was sure that it was not a software fault (because the same one worked on KC705 board), it meant that hardware mistakes were present.

Taking a deeper look to the board, it was noticed that the Ethernet driver IC, which is the module implementing the link between the FPGA and the Ethernet connector itself,

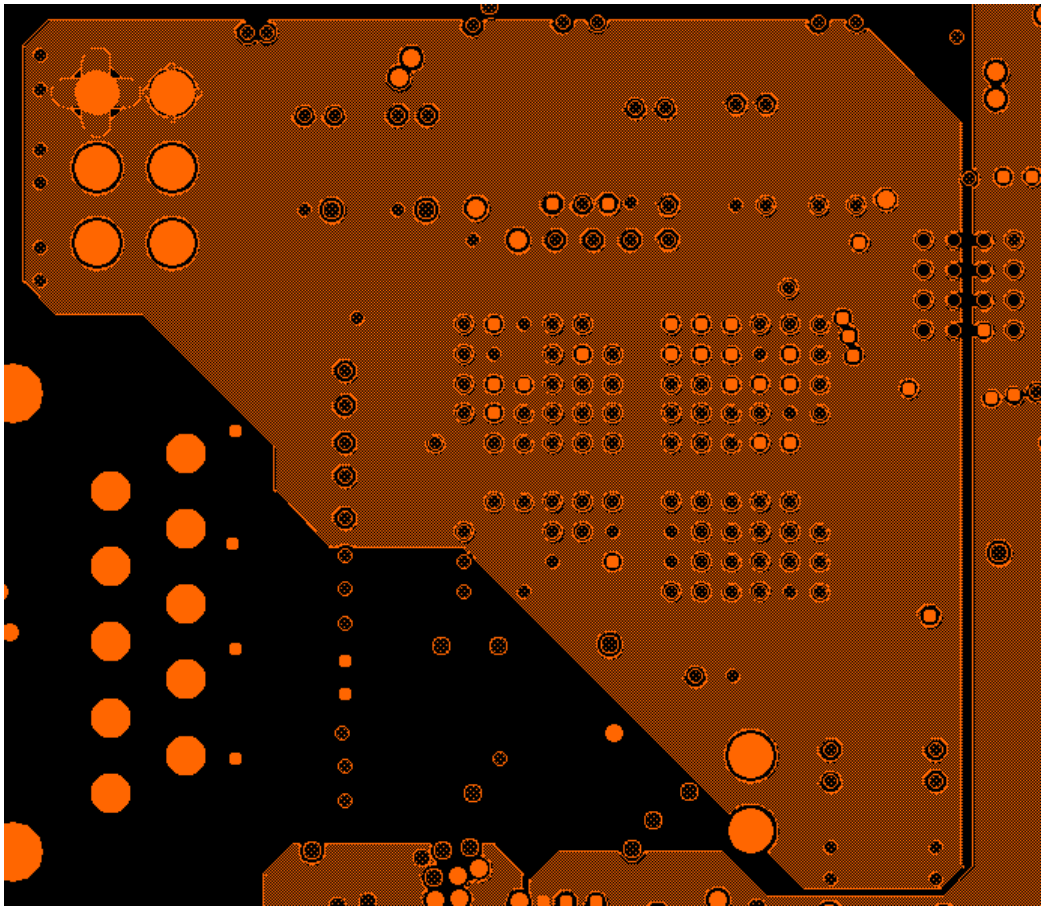
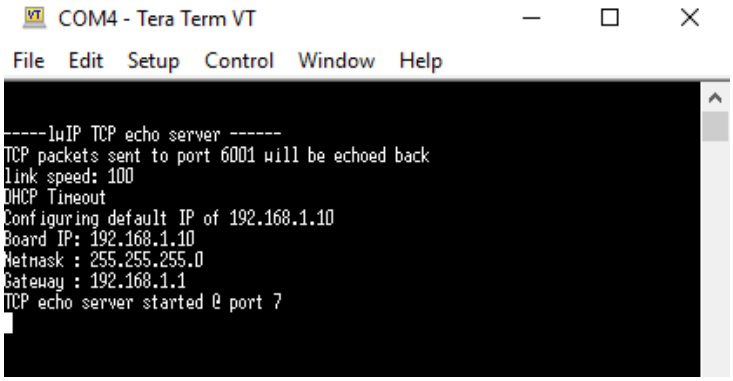


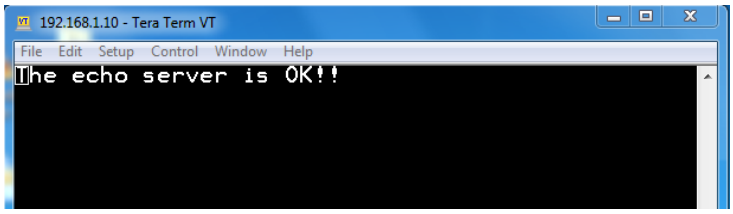
Figure 3.22: Detail of the bug affecting the Pixel-ROD board. The closed orange area is the power supply plane of the Ethernet driver IC, clearly not connected to any power source.

was not powered. By means of the PCB viewer the answer was found: the closed orange area shown in Figure 3.22 represents the power supply plane for the Ethernet driver IC, that is clearly not connected to any power source, except for the configuration header in the top left corner, which evidently can not be a power input. By means of this very header, a wire supplying the needed voltage was connected to the plane, instantly fixing the problem. In fact, after this patch was applied, the software application immediately started working, producing the result reported Figure 3.23.

As shown in Figure 3.23a, the application starts an echo server with precise parameters, such as IP, link speed, server port and protocol. By connecting another terminal to IP just set, it has been possible to communicate keyboard inputs to the server, which



(a) *UART terminal response during echo server test application run.*



(b) *Result of keyboard input to echo terminal.*

Figure 3.23: Echo server test results.

then responds printing on the terminal an output identical to the input, as shown in Figure 3.23b. This result validated the Ethernet functionality, along with the UART one as well, which has provided a terminal where the Microblaze printed the results of the running application.

3.4 SFP to GBTx test

The most significant test that has been performed on the Pixel-ROD board is the one involving the fast communication through the SFP port. This test is of great importance in many ways: firstly, it actually uses the only port supporting optical link connection, thereby reaching the fastest data rate available now; secondly, in this test the Pixel-ROD has been interfaced with other boards, hence providing an environment that resembles better the one expected form this kind of boards; last but not least, in this test the Pixel-ROD has been interfaced with a *GBTx* Application Specific Integrated Circuit (ASIC), which is a radiation-hard Integrated Circuit (IC) specifically intended to perform data

acquisition for physical experiments with high level of radiation, like ATLAS or any other experiment at LHC. In order to provide a better understanding of the test described in

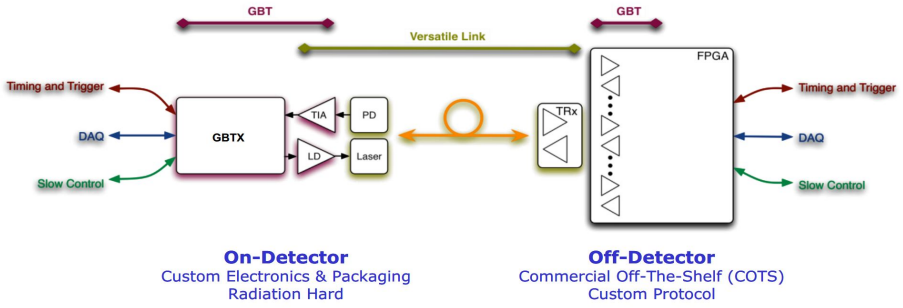
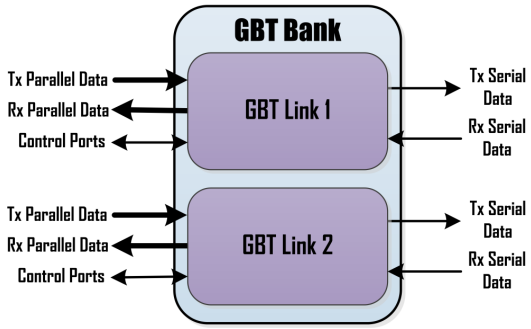


Figure 3.24: GBT link architecture.

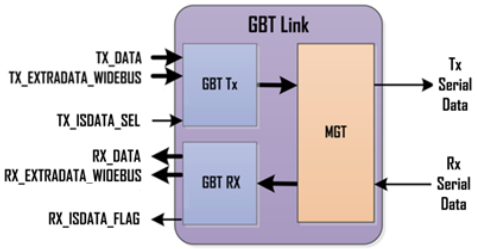
this section, it is necessary to explain what is GBTx. The GBTx ASIC is a radiation tolerant chip that can be used to implement multi-purpose high speed (nominal bandwidth of 4.8 Gbps, user range bandwidth from 3.2 Gbps up to 4.48 Gbps [24]) bidirectional optical links for high-energy physics experiments. As Figure 3.24 shows, this link logically provides three different data paths for TTC, Data Acquisition (DAQ) and Slow Control (SC). However, these three logical paths do not need to be physically separated and therefore are merged into a single optical link. The clear aim of such architecture is to provide a single, bidirectional (two fibres), optical, point-to-point link that can provide very high reliability in the harsh radiation environment typical of high energy physics at LHC. Since not only the integrated components but also the a portion of the linking fibres are placed in a high radiation environment, the choice of the appropriated fibres and opto-electronics components has to be accurate. This is what the *Versatile Link* in Figure 3.24 refers to. On the other side, the Versatile Link connects on-detector electronics to off-detector electronics: the former is made by custom developed components (like GBTx ASIC) that can withstand the harsh environment, while the latter is made by Commercial-Off-The-Shelf (COTS) components, in order to take advantage of the latest commercial technologies, enabling efficient data concentration and data processing from many front-end sources to be implemented in very compact and cost efficient trigger and DAQ interface systems. In order to implement the communication between the GBTx ASIC and Kintex FPGA (which controls the SFP port) it is necessary to instantiate an equivalent version of the architecture of the GBTx ASIC on the Kintex-7

itself: therefore, this FPGA-based architecture is named *GBT-FPGA core*. In order to

(a) *GBT FPGA bank.*



(b) *Detail of GBT FPGA Bank, with GBT links.*

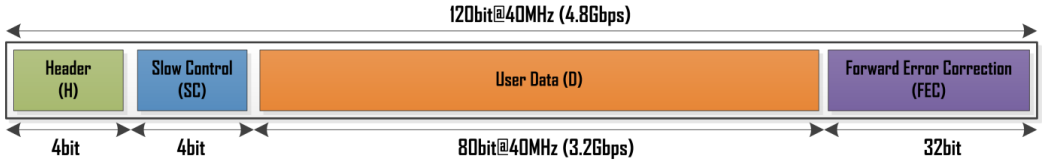


(c) *Detail of GBT FPGA Bank, with a single GBT link.*

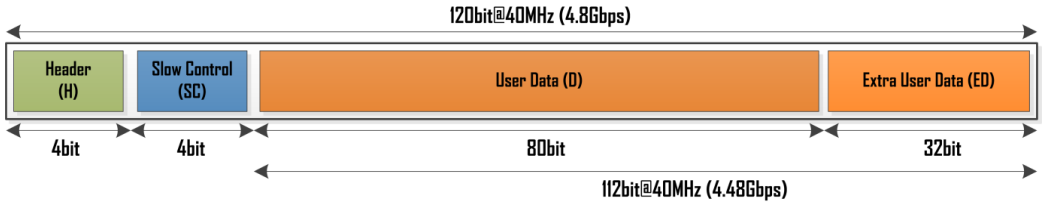
Figure 3.25: GBT FPGA core block diagram representation

ease the in-system implementation and the user support of the GBT-FPGA core, all of its components are integrated in a single module called *GBT Bank*. This module, as shown in Figure 3.25a, is composed by: a *GBT Tx/Rx* to receive and transmit parallel data; a Multi-Gigabit Transceiver (MGT) to receive and transmit serial data; finally, an input/output clocking signal, in order to provide the needed synchronization for this high-speed signals. As reported in Figure 3.25b, each GBT Bank can support more than one *GBT link*, in order to provide a stream flow for the data. Therefore, each GBT link takes as input parallel data, which are then serialized by means of the built-in GTX transceivers of the FPGA, in order to provide the Tx flow reported in Figure 3.25b. The same data path can be covered in the backward direction, thereby taking as inputs serial data, which are de-serialized by FPGA's GTX transceivers and transmitted as Rx parallel data. It may appear that the GBT bank adds nothing to the FPGA logic, since the all the work is done by the built-in transceivers of the FPGA: clearly, this is not the case and Figure 3.25c helps understand the reason. When parallel data enter as input in the GBT link, in the first place, they pass through the GBT Tx block: this first stage

has the task to prepare data to be serialized by the transceivers and is where the GBT logic takes place. Indeed, the MGT block comprises the transceivers themselves and has the task to serialize the data.



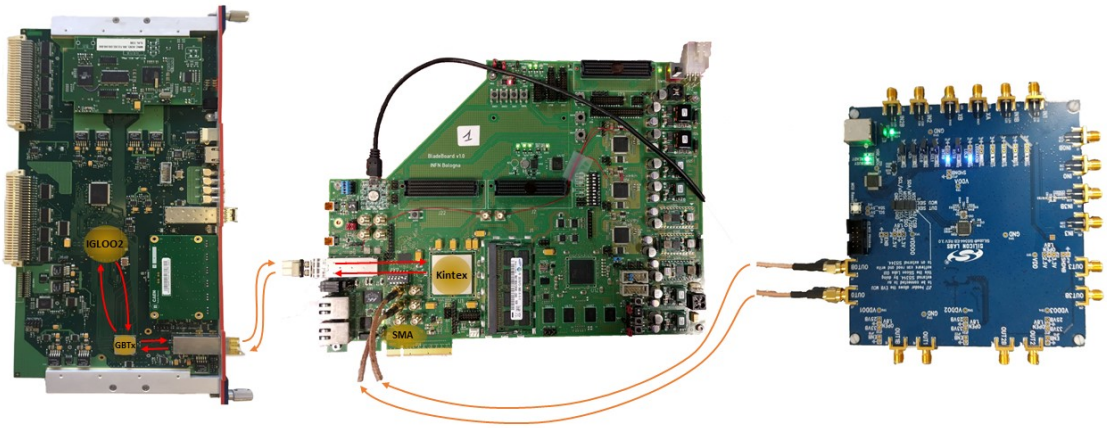
(a) *The GBT-Frame encoding frame.*



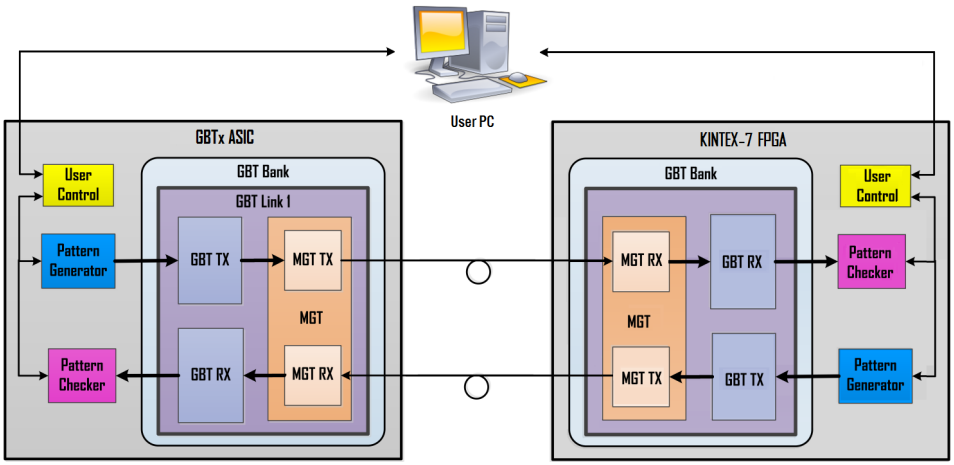
(b) *The Wide-Bus encoding frame.*

Figure 3.26: Schematic description of the two possible GBT encoding frames.

The GBT Tx and GBT Rx blocks main purpose is to encode the data into one of two possible frame formats: the *GBT-Frame* or the *Wide-Bus* frame. As Figure 3.26 shows, both frames implements a 4 bit Header and a 4 bit section for Slow Control; however, the GBT-Frame (see Figure 3.26a) supports a 80-bit-wide user data word, followed by the 32-bit Forward Error Connection (FEC), which implements the ability to find and correct burst of bit errors caused by a Single Event Upset (SEU), like the interaction of a transmitted bit with a radiation charged particle. Given this resistance to radiation, this type of encoding can be used for DAQ, TTC and Experiment Control (EC) purposes. The nominal transmission data rate is of 4.8 Gbps, since 120-bit-wide words are sent at with a frequency of 40 MHz; however, since in this type of frame only 80 bits are user-defined for data transmission, the real data rate lowers to 3.2 Gbps (80-bit-wide words at 40 MHz). The Wide-Bus frame (see Figure 3.26b) eliminates these constraints, used to catch and fix errors, by replacing the 32-bit-wide FEC part of the frame with Extra user Data (ED), speeding up the data rate to 4.48 Gbps. The test set-up is shown in Figure 3.27a, where the Pixel-ROD board has been connected to other two boards. The one the left is the Digital Readout Module v2 (DRM2), used in ALICE TOF experiment, that implements the GBTx ASIC, as well as IGLOO2 FPGA by Microsemi; the one on



(a) *Brief view of the set-up used for SFP to GBT test. In orange are highlighted the optical links to and from Pixel-ROD, while the actual path of GBT-frame data is highlighted in red.*



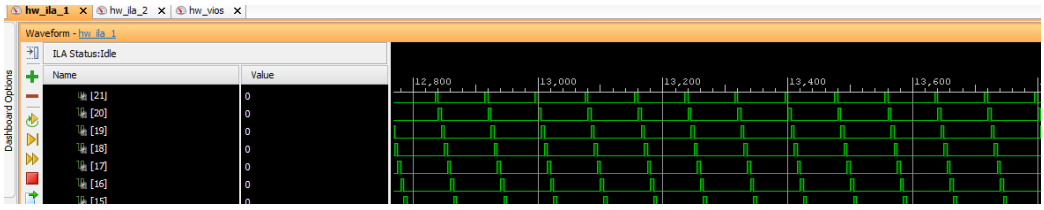
(b) *Simplified block diagram of GBTx to SFP test interconnections between FPGAs.*

Figure 3.27: Test set-up for GBTx to SFP test.

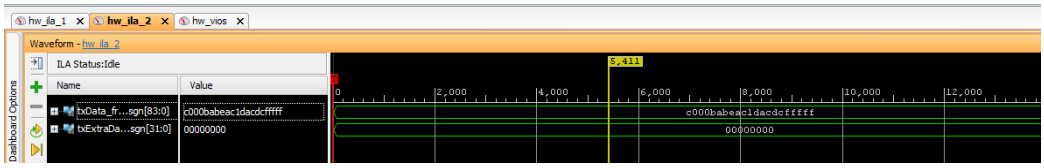
the right is the evaluation board of Si-5344 Phase-Locked Loop (PLL) by Silicon Labs, which is used in this test in order to provide the a low-jitter 120 MHz differential clock to the Kintex. This 120 MHz clock source is used as reference clock for GTX transceivers on Pixel-ROD and is received on the dedicated *SubMiniature version A* (SMA) connectors

available on the board.

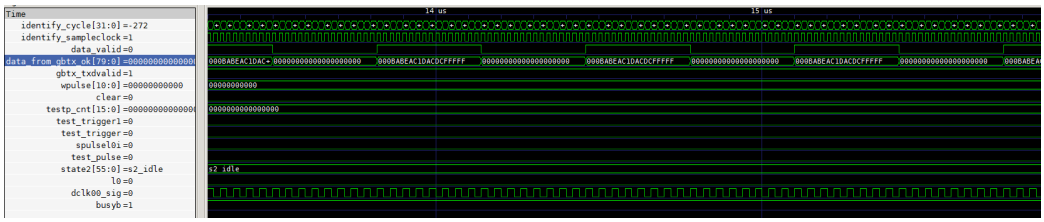
In this test, through the Vivado software, the Pattern Generators reported in Figure 3.27b were programmed to send two different patterns: the one implemented on Kintex FPGA was creating a constant word; the one on IGLOO2 FPGA was implementing a *Walking Ones* pattern, which is obtained by setting to zero all the User Data bits and then moving, from the Least Significative Bit (LSB) to the Most Significative Bit (MSB), a single logical 1. This test was performed using a GBT-Frame, since this frame is the one that will be used in high energy physics experiments, which are the targeted tasks for this board. The sent patterns were then received on the other FPGA and controlled by the Pattern Checker (see Figure 3.28a). The results of this test proved to be successful, since both patterns were correctly received, as it is shown in the Figure 3.28. This test is important not just because it shows that the SFP can work at half of its expected data rate (9.8 Gbps), but also because it is performed with protocols, devices and boards that are already used in high-energy physics experiments at CERN, such as ALICE.



(a) *Result of received Walking Ones pattern on the Kintex-7.*



(b) *Constant word sent by Kintex FPGA.*



(c) *Result of the received constant word on IGLOO2 FPGA.*

Figure 3.28: Result acquired with Vivado software of the GBT test.

3.5 Zynq interfaces and memory test

Finally, the last test that was performed aimed to validate the interfaces (UART and Ethernet) and the memory modules for the Zynq FPGA. Since the scope of the test is the same as the one presented in 3.3, the procedure that has been used is very similar too. However, two significant differences made this test more challenging: firstly, we could not try neither the firmware nor the software on the demo board, since we did not own it; secondly, it was the first time that the ARM CPU was used, which was for us an almost completely new device.

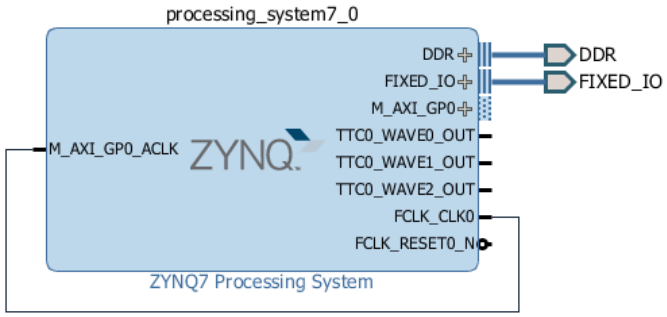


Figure 3.29: The Zynq Processing System IP core as shown on Vivado. On the right side of the core, the *DDR* and *FIXED_IO* ports implement the connections between the PS itself and the actual interfaces and memories.

On the other side, the firmware that had to be used in this test was, for its own nature, very simple: using Vivado, only one major IP core needed to be instantiated, which was the *Zynq processing System* [25], as shown in Figure 3.29. This IP core simply instantiates and connect the Zynq PS to devices and memory present on the board. This IP then includes every other block shown in Figure 3.19, as Figure 3.30 reports. Through the customization window is possible to activate and configure the interfaces which needs to be tested; in fact, as it is possible to notice in the upper left side of Figure 3.30, some interfaces are ticked, such as the UART and the Ethernet, meaning that they are configured within the Zynq PS IP core.

Once the firmware was developed, the two software applications described in the section 3.3 were prepared in SDK for this project as well. However, at first, launching them on the Zynq did not show any result, as the process was stopping before coming to com-

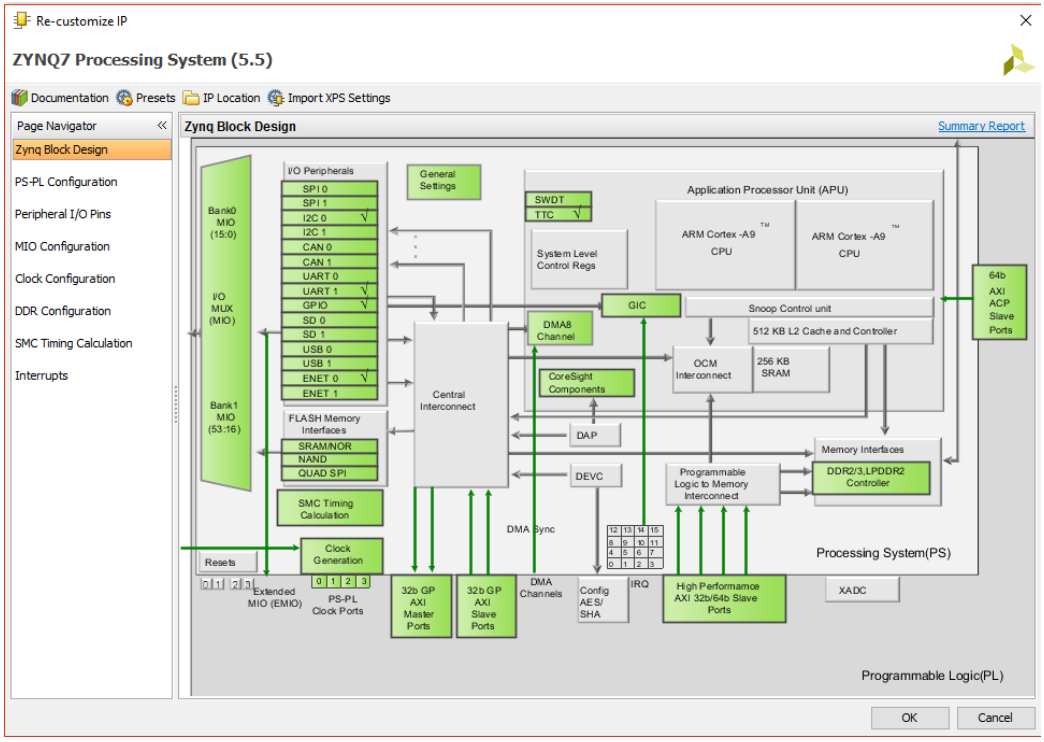


Figure 3.30: Configuration window for Zynq Processing System, showing in green all the activated blocks.

pletion.

An accurate analysis, started by evaluating the correctness of each pin configured on the FPGA, brought the attention to the switch, named *SWZ16*, which is shown in Figure 3.31. This switch is very important, since it sets from where the Zynq loads its configuration: in our case this switch was set to load the configuration from an external empty memory, thus blocking our attempts to use the Zynq. Once the configuration of the switch was fixed, setting it so that the Zynq could load from JTAG, the applications started to run as expected.

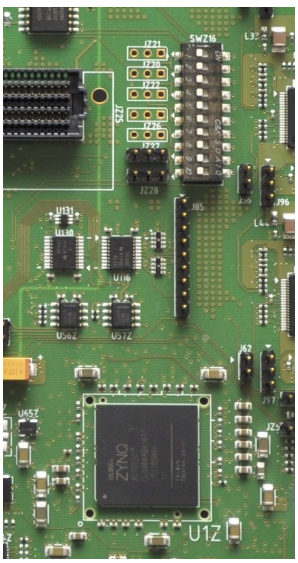
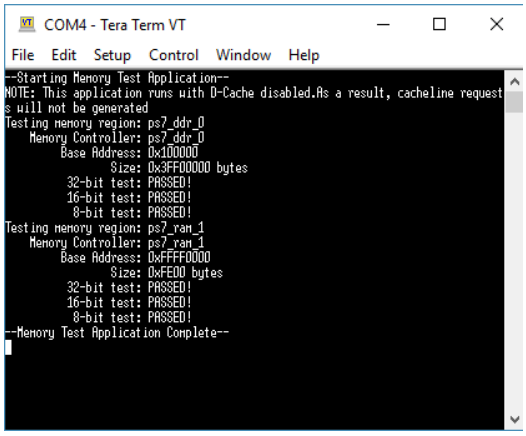


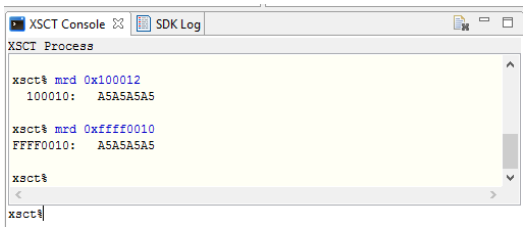
Figure 3.31: The Zynq with the 10-ways SWZ16 switch on top.

For what concerns the memory test application, it runs just as described in the paragraph 3.3.2, with just a slight difference, which is shown in Figure 3.32a: since

the Zynq PS already features an internal RAM memory (see paragraph 2.2.2), the test checks this memory region as well, which is referred as *ps7_ram_1*, in contrast with *ps7_ddr_0*, that is the actual DDR3 under test. By means of XSCT terminal a *mrd* memory read command was issued to the Zynq, targeting addresses both on the internal RAM and on the DDR3 components: each read transaction is displayed in Figure 3.32b, reporting the 8-bit-wide word *A5* that has been written during the test, thereby assessing the success of the it and validating the DDR3 components. Finally, the Ethernet application was run as

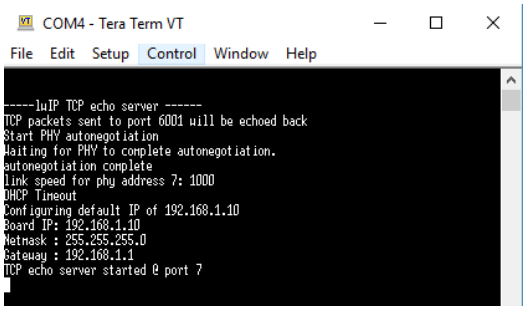


(a) *Test response on UART terminal.*

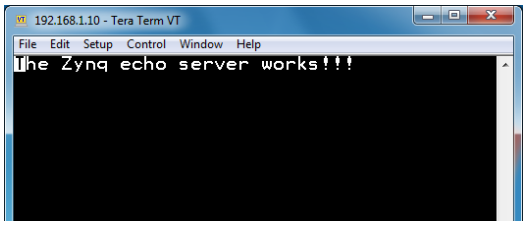


(b) *Response to read operation on Zynq's tested memory.*

Figure 3.32: Result of memory test performed on Zynq.



(a) *Ethernet test response on UART terminal.*



(b) *Result of keyboard inputs on echo server terminal.*

Figure 3.33: Results of Zynq's echo server test.

well, thereby creating an echo server on Zynq PS, whose configuration parameters were

reported on the UART terminal, as shown in Figure 3.33a. Again, by connecting another terminal to the echo server using the TCP/IP protocol, it has been possible to give keyboard input to the server; on the other side, the server replied to each input by sending it back to the terminal, thus creating the example output shown in Figure 3.33b.

With this test completed as well, we could assess that both the Ethernet port and the DDR3 memory were correctly validated, as well as the UART port, to which a terminal has been connected during both application runs, thereby assessing the correct functionality of the port.

Conclusions and future developments

The work described in this thesis describes a new project in the field of off-detector electronics for high-energy physics experiments. Taking advantage from the experience acquired during the IBL project on the readout systems for ATLAS Pixel Detector, it was decided to develop a new electronic board, named Pixel-ROD, to increase the data acquisition performances on such system. Therefore, at first, my contribution on this project has involved the completion of the board design development; successively, I have been focused on more critical tasks, such as debug and validation of the firsts prototypes, creating dedicated tests to achieve these objectives.

At present - March 2017 - two prototypes have been produced and almost all of their interfaces and devices have been successfully debugged and tested here in Bologna, in the Electronic Design Laboratory of INFN (Istituto Nazionale di Fisica Nucleare) and DIFA (DIpartimento di Fisica e Astronomia). Further tests need to be created and performed in order to assess a complete validation of both prototypes. Nevertheless, the Pixel-ROD board has already shown, during the tests, its capability to perform fast data acquisition and processing. Therefore, future tests will aim to integrate Pixel-ROD within a simple data acquisition chain, that implements the well-known standard from ATLAS Pixel Detector or even the AURORA protocol from CMS experiment.

In order to guarantee faster electronics, new boards for the data acquisition systems need to be developed, featuring the most recent devices and interfaces, that can grant the desired performances. In the high-energy physics community, so far only few high performance electronic boards have been presented and validated. The Pixel-ROD board represents an important contribution offered to the international CERN community for the future experiments.

In conclusion, looking towards the LHC upgrades and beyond, when the experiments will need faster and more efficient electronics, this board, or its possible future upgraded

revisions, might be part of an electronic readout chain for tracking systems.

Ringraziamenti

Molte sono le persone che hanno contribuito al raggiugimento di questo mio traguardo, ciascuna delle quali ha fornito un aiuto fondamentale.

Ringrazio anzitutto il professor Alessandro Gabrielli, Relatore, ed il dottor Davide Falchieri, Correlatore, che mi hanno fornito supporto e guida, senza i quali questa tesi non esisterebbe; ringrazio anche Giuliano Pellegrini, la cui pazienza mi ha guidato all'inizio del progetto qui descritto.

Un ringraziamento particolare va poi a tutti gli amici, i quali hanno condiviso con me questo percorso di studi, allietandolo ed incoraggiandomi.

Infine vorrei ringraziare tutta la mia famiglia e la mia ragazza, veri pilastri portanti su cui mi sono poggiato durante tutta questa lunga avventura.

Bibliography

[1] CERN's website, `https://home.cern/`

[2] ATLAS website, `http://atlas.cern/`

[3] The ATLAS Collaboration, *ATLAS detector and physics performance Technical Design Report*, vol. 1, ATLAS TDR 14, CERN/LHCC 99-14, 25 May 1999, CERN, Geneva.

[4] ATLAS IBL Community, *Insertable B-Layer Technical Design Report*, ATLAS TDR 19, CERN/LHCC 2010-013, 15 September 2010, CERN, Geneva.

[5] G. Balbi, D. Falchieri, A. Gabrielli, L. Lama, R. Travaglini, S. Zannoli, *IBL ROD board rev C reference manual*, November 2012, `https://espace.cern.ch/atlas-ibl/OffDecWG/`

[6] Béjar Alonso I. et al., *High-Luminosity Large Hadron Collider (HL-LHC) Preliminary Design Report*, CERN-2015-005, 17 December 2015, CERN, Geneva.

[7] J. Anderson, K. Bauer, A. Borga, H. Boterenbrood, H. Chen, K. Chen, G. Drake, M. Dönszelmann, D. Francis, D. Guest, B. Gorini, M. Joos, F. Lanni, G. Lehmann Miotto, L. Levinson, J. Narevicius, W. Panduro Vazquez, A. Roich, S. Ryu, F. Schreuder, J. Schumacher, W. Vandelli, J. Vermeulen, D. Whiteson, W. Wu and J. Zhang, *FELIX: a PCIe based high-throughput approach for interfacing front-end and trigger electronics in the ATLAS Upgrade framework*, 2016 JINST 11 C12023, TWEPP 26-30 September 2016, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany.

[8] Xilinx®, *KC705 Evaluation Board for the Kintex-7 FPGA*, 26 August 2016, https://www.xilinx.com/support/documentation/boards_and_kits/kc705/ug810_KC705_Eval_Bd.pdf

[9] Xilinx®, *Spartan 6 Family Overview*, DS160 (v2.0), 25 October 2011, https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

[10] Xilinx®, *7 Series FPGA Overview*, DS180 (v1.17), 27 May 2015, https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

[11] Vita's website, http://www.vita.com/fmc

[12] PCI-SIG®, *PCI Express Base Specification*, Revision 2.1, 4 March 2009.

[13] Xilinx®, *ZC702 Evaluation Board for the Zynq-7000 XC7Z020 All Programmable SoC*, UG850 (v1.5), 4 September 2015, https://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/ug850-zc702-eval-bd.pdf

[14] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, R. W. Stewart, *The Zynq Book Embedded Processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC*, First Edition, Strathclyde Academic Media, 2014.

[15] Xilinx®, *Zynq-7000 All Programmable SoC Technical Reference Manual*, UG585 (v1.10), 23 February 2015, https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

[16] L. Lama, G. Balbi, D. Falchieri, G. Pellegrini, C. Preti, A. Gabrielli, *A PCIe DAQ board prototype for Pixel Detector in High Energy Physics*, 12-01-C01073, Journal of Instrumentation, 2017.

[17] Texas Instrument website, http://www.ti.com/product/UCD9248

[18] System Management Interface Forum, *PMBus™ Power System Management Protocol Specification Part II - Command Language*, Revision 1.2, 6 September 2010.

[19] ARM®, *AMBA® AXI™ and ACE™ Protocol Specification*, Issue D, 28 October 2011.

[20] Xilinx®, *MicroBlaze Processor Reference Guide*, UG984 (v2016.4), 30 November 2016, https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_4/ug984-vivado-microblaze-ref.pdf

[21] Xilinx®, *AXI Ethernet Subsystem*, PG138 (v6.2), 1 October 2014, https://www.xilinx.com/support/documentation/ip_documentation/axi_ethernet/v6_2/pg138-axi-ethernet.pdf

[22] Xilinx®, *AXI Traffic Generator*, PG125 (v2.0), 6 April 2016, https://www.xilinx.com/support/documentation/ip_documentation/axi_traffic_gen/v2_0/pg125-axi-traffic-gen.pdf

[23] Xilinx®, *AXI UART Lite*, PG142 (v2.0), 5 October 2016, https://www.xilinx.com/support/documentation/ip_documentation/axi_uartlite/v2_0/pg142-axi-uartlite.pdf

[24] S. Baron, J. Mendez, *GBT FPGA User Guide*, version 1.3, 13 April 2016.

[25] Xilinx®, *Vivado Design Suite User Guide - Embedded Processor Hardware Design*, UG898 (v2016.3), 30 November 2016, https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_1/ug898-vivado-embedded-design.pdf