UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira - SP

LUCAS ARRUDA RAMALHO

An FPGA based 3.8 Tbps Data Sourcing and Emulator System

Ilha Solteira - SP
2018

LUCAS ARRUDA RAMALHO

An FPGA based 3.8 Tbps Data Sourcing and Emulator System

Tese apresentada à Faculdade de Engenharia do Câmpus de Ilha Solteira - UNESP como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica.

Especialidade: Automação.

Prof. Dr. Aílton Akira Shinoda
Orientador

Ilha Solteira - SP
2018

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

R165f

Ramalho, Lucas Arruda.
A FPGA based 3.8 Tbps data sourcing and emulator system / Lucas Arruda Ramalho. -- Ilha Solteira: [s.n.], 2018
106 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2018

Orientador: Aílton Akira Shinoda
Inclui bibliografia

1. LHC. 2. CMS. 3. Outer tracker. 4. ATCA. 5. FPGA. 6. High speed serial link protocol.

João Josué Barbosa
Serviço Técnico de Biblioteca e Documentação
Diretor Técnico
CRB 8-5642

unesp

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA TESE: An FPGA based 3.8 Tbps Data Sourcing and Emulator System

AUTOR: LUCAS ARRUDA RAMALHO
ORIENTADOR: AILTON AKIRA SHINODA

Aprovado como parte das exigências para obtenção do Título de Doutor em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO pela Comissão Examinadora:

Prof. Dr. AILTON AKIRA SHINODA
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. JEAN MARCOS DE SOUZA RIBEIRO
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. ROGÉRIO LUIZ IOPE
Núcleo de Computação Científica / Universidade Estadual Paulista Júlio de Mesquita Filho

Prof. Dr. ANTONIO VILELA PEREIRA
Departamento de Física / Universidade do Estado do Rio de Janeiro - UERJ

Prof. Dr. GILVAN AUGUSTO ALVES
Departamento de Física de Altas Energias / Centro Brasileiro de Pesquisas Físicas - CBPF

Ilha Solteira, 23 de fevereiro de 2018

À minha família, em especial aos meus pais Luiz Leôncio e Zildineti, ao meu irmão Douglas Ramalho, por todo amor, apoio, confiança e incentivo em todos os momentos.

ACKNOWLEDGMENTS

First of all, I would like to thanks my father (Luiz), my mother (Zildineti), my aunt (Jane), my brother (Douglas), my sister in law (Adriana), and my nephew (João). They represents a small part of my huge family. Everytime, I had a difficult time, you were there to give me support and cheer for me. I insert in the same package of contributions all my friends from my hometown. For sure, great part of the desire to return home, it is to stay closer to you all.

I would like to thanks professor Aílton Akira Shinoda by being my advisor since my undergrad that was performed at Cuiabá. His advices were very important for my growing in cientific methods and also interpersonal relationship inside those collaborations that we participate nowadays. As a professional, I owe to you all oportunities I had, like to do the masters degree, to do the PhD, to travel to US, to become a professor like you.

In the same way, I thanks to all professors I had in the PhD program at UNESP Ilha Solteira were important. Represented by professor Nobuo, that gave me advices when my mind was a little foggy. I thanks the whole PhD program.

Of course, I thanks all friends I did during the program. Thanks to Cássia, Douglas, Gilvani and Rothschild representing all community of Mato Grosso students. I think we really made a difference at Ilha Solteira, and we let a good scar in the PhD program forever. This travel started a long time ago, and without the friendship from you all the would had become more difficult for sure.

Representing the other friends I met in Ilha, I thanks Ortega, Miller, Flavilene and Patricia to push my dedication in the work to increase or even the parties when we needed. I hope this fellowship does not end with the PhD, despite the distance.

My professionals habilities, and personal experience as well, were enhanced by the experiences I had during my PhD. Being a network computer guy from the middle of Brazil and to have a chance of work with instrumention in high energy physics was only possible thanks to SPRACE group led by Prof. Sergio Novaes. I thanks him as SPRACE representative that allowed me to make part of one of the most important HEP and HPC research group of Brazil.

This experience, that started at SPRACE learning about accelerators, detectors and standard model, and was transformed in a adventure for me when our group started to collaborate with Fermilab and the L1 trigger proposal. I had a chance to travel to US and to learn a lot at

Fermilab with the whole AM+FPGA collaboration. So, I also thanks Ted Liu, Jamieson and Marco Trovato, that worked closer to me and represent all other Fermilab people that helped.

The contributions did at Fermilab were possible thanks to the SPRACE teams assembled at instrumentation laboratory. People like, Thiago Paiva, Mario Vaz, Vitor Finotti, André Cascadan, Rogério Iope, among others, were working closely with me to implemente several aspects of the ATCA infrastructure. Thanks guys, we did a nice job.

Still talking about Fermilab, I need to thank all the Brazilian and non-Brazilian friends I made in Fermilab. You were all part of making my time away from home, and Brazil, not boring. And special thanks to O'Sheg, that reminded me to always follow the light even when our hearth tends see darkness in the problems we face. O'Sheg, during my time at Fermilab your words and support were far more important than you can imagine.

Finally, despite to be the most important, I thanks God to give the oportunity to do this PhD in my life. You gave me health, despite all BBQ's, to be in the right place, in the right moment, and with capacity to enjoy all the path you prepared for me.

"Don't forget to breathe."

The Silent Monk, The Forbidden Kingdom (2008)

RESUMO

A evolução dos Multi Gigabit Transceivers (MGT) nos Field Programmable Gate Arrays (FPGA) trouxeram oportunidades para o desenvolvimento de sistemas de aquisição e formatadores de dados em diversas áreas. As novas famílias de FPGAs são capazes de lidar com canais de transmissão com velocidade da ordem de Gbps que utilizam protocolos seriais de alta velocidade, podendo assim se tornar o futuro dos processadores downstream ou upstream. Os sistemas digitais criados para esse propósito, precisam ser confiáveis e síncronos entre dezenas de canais e placas. Como forma de permitir o teste de projetos com essa taxa massiva de bits, essa tese descreve o desenvolvimento do Data Sourcing System (DSS). Esse sistema deve ser capaz de testar qualquer application upstream ou downstream, permitir controle e acesso remoto aos sinais internos dos FPGAs, medir sincronismo e latência entre MGTs e avaliar integridade de links através de bit error rate (BER). Este trabalho faz parte de uma colaboração internacional liderada pelo Fermilab que propôs, com a contribuição do sistema descrito nesta tese, um sistema de trigger de nível 1 para o Compact Muon Solenoid (CMS) Outer Tracker. O dectetor CMS é um experimento vinculado ao European Organization for Nuclear Research (CERN). O DSS foi implementado sobre a placa Pulsar 2b, uma placa padrão Advanced Telecommunication Computing Architecture (ATCA), desenvolvida pelo Fermilab, que conta com um dispositivo FPGA para programação e costumização de aplicações. O setup de hardware utilizado foi construído sobre dois bastidores ATCA com 12 placas Pulsares 2b em cada. A taxa de dados máxima atingida foi de 3.84 Tbps entre os dois bastidores ATCAs. O DSS está operacional e foi utilizado para emular o fluxo de dados de saída do CMS Silicon Outer Tracker, e auxiliar na demonstração da proposta trigger de nível 1. Esta tese descreve essa demonstração como estudo de caso, que testa o formatador de dados do trigger (downstream) através do DSS emulando a saída de dados do detector. Nesse estudo de caso, tanto o DSS e o trigger proposto foram implementados utilizando o mesmo hardware ATCA e a Pulsar 2b. O foco do estudo de caso é descrever a comunicação entre o Data Sourcing shelf e o Pattern Recognition shelf. O DSS atendeu aos requisitos da demonstração provendo uma interface de usuário que permite aos desenvolvedores de trigger inserir sinais de controle e executar operações de leitura e escrita de forma remota nos FPGAs.

Palavras-chave: LHC. CMS. Outer Tracker. ATCA. FPGA. Protocolos Seriais de Alta Velocidade. Processamento de Dados. Conceitos e Sistemas de Trigger. Sistema de Controle de Detector. Sistemas de controle e monitoramento online.

ABSTRACT

The evolution of Fiel Programmable Gate Array (FPGA) Multi Gigabit Transceivers (MGT) brought opportunities for data formatter and data acquisition projects in several areas. The newer FPGA families are capable of handling Gigabits per second (Gbps) I/Os implemented using high speed serial link protocols and to become the future downstream processors. The digital systems created for that purpose need to be reliable and synchronous between dozens of channels and boards. To allow the test of such massive bitrate projects, this work implemented the Data Sourcing System (DSS) e- mulator that is able to produce synchronized data in 12 boards, 480 channels, delivering up to 8 Gbps for each of them. This work is part of a international collaboration, led by Fermilab, that proposed with the contribuition of the system described in this thesis, a Level 1 (L1) tri- gger for the Compact Muon Solenoid (CMS) Outer Tracker. The CMS detector is an European Organization for Nuclear Research (CERN) experiment. The DSS is based on the Pulsar 2b, a custom Advanced Telecommunication Computing Architecture (ATCA) standard FPGA-based board designed by Fermilab to be a scalable high speed link processor system. This hardware setup was implemented at Fermilab using two interconnected ATCA shelves with 12 Pulsar 2b on both. The results show that the system is able to provide data at 3.8 Terabits per second (Tbps), and to measure synchronization, latency and bit error rate of the MGTs. The system is operational and was already used to emulate the CMS Silicon Tracker data, and helped the demonstration of a L1 Trigger approach. This thesis describes the demonstration performed as case of study, which used the DSS as upstream system and tested the trigger data delivery as a downstream. In the case of study, both DSS and the proposed trigger are performed by the same ATCA hardware and the Pulsar 2b. The case of study focused to describe the communication between the Data Sourcing shelf and the Pattern Recognition shelf. Data Sourcing reached those requirements for the demonstration and provided a user interface that allows the trigger developers to insert control signals or to perform W/R operations inside Pulsar 2b FPGA block memories.

Keywords: LHC. CMS. Outer Tracker. ATCA. FPGA. High Speed Serial Link Protocol. Data Processing. Trigger concepts and systems. Detector control systems. Control and monitor systems online.

FIGURE LIST

TABLE LIST

ACRONYMS

ADC	Analog to Digital Converter
AGATA	Advanced GAmma Tracking Array
ALICE	A Large Ion Collider Experiment
AM	Associative Memory
AM+FPGA	AM plus FPGA
AMC	Advanced Mezzanine Cards
ASIC	Application-Specific Integrated Circuit
ATCA	Advanced Telecommunication Computing Architecture
ATLAS	A Toroidal LHC ApparatuS
BC	BX Counter
BC0	Bunch Crossing Zero
BE	Back End
BER	Bit Error Rate
BRAM	Block RAM
BX	Bunch Crossing
CBC	CMS Binary Chip
CBM	Compressed Baryonic Matter
CCD	Charge-Coupled Device
CDF	Cumulative Probability Function
CERN	European Organization for Nuclear Research
CIC	Concentrator Integrated Circuit
CMS	Compact Muon Solenoid
DAQ	Data Acquisition
DC	Direct Current
DFE	Decision Feedback Equalizer
DS	Data Sourcing
DSB	DS Board
DSM	DS Module
DSP	Digital Signal Processor
DSS	DS System
DTC	Data Trigger and Control
EC0	Event Counter Zero
ECAL	Eletromagnetic Calorimeter

FE	Front End
FIFO	First-In-First-Out
FMC	FPGA Mezzanine Card
FNAL	Fermi National Accelerator Laboratory
FPGA	Field Programmable Gate Array
FRU	Field Replaceable Unit
FSM	Finite State Machine
FW	Firmware
Gbps	Gigabits per second
GeV	Gigaeletrovolt
GPU	Graphics Processing Unit
GTH	Gigabit transceiver version H
HCAL	Hadronic Calorimeter
HEP	High Energy Physics
HL-LHC	High Luminosity LHC
HPM	Hardware Platform Managemen
HV	High Voltage
I/O	Input/Output
I2C	Inter-Integrated Circuit
IBERT	Integrated Bit Error Ratio Tester
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IP core	Intellectual Property core
IPM	Intelligent Platform Management
IPMB	IPM Bus
IPMB-L	IPMB - Local
IPMC	IPM Controller
IPMI	IPM Interface
IP-Sec	IP Secure
ISI	Inter-Symbol Interference
JET	Joint European Torus
JTAG	Joint Test Action Group
L1	Level 1
L1 Accept	L1 Accept
L1TT	L1 Tracking Trigger

LBNL	Lawrence Berkeley National Laboratory
LHC	Large Hadron Collider
LHCb	LHC beauty
LPC	LHC Physics Center
LP-GBT	Low Power Gigabit Transceiver
LPM	Low-Power Mode
LV	Low Voltage
MGT	Multi Gigabit Transceiver
MMC	Module Management Controller
m-RX	Multi Receivers
m-TX	Multi Transceivers
NASA	National Aeronautics and Space Administration
PA	Port A
PB	Port B
PC	Personal Computer
PCB	Printed Circuit Boards
PCIe	Peripheral Component Interconnect Express
PCS	Physical Coding Sublayer
Physlots	Physical Slots
PICMG	PCI Industrial Computer Manufacturers Group
PLL	Phase Lock Loop
PMA	Physical Medium Attachment
PRB	Pattern Recognition Board
PRBF1	Pattern Recognition Board Format version 1
PRBF2	Pattern Recognition Board Format version 2
PRBS	Pseudo Random Binary Sequence
PRS	Pattern Recognition Shelf
PS	Power Source
PU	Pile Up
QSFP+	Quad Small Form-factor Pluggable Plus
R&D	Research and Development
RAM	Random Access Memory
RIO	RapidIO
RTM	Rear Transition Module
RX	Receiver
SPRACE	São Paulo Research and Analysis Center
Tbps	Terabits per second
TCP	Transmission Control Protocol

TCP/IP	TCP over IP
TCS	Trigger Control System
TDPBRAM	True Dual Port BRAM
TDR	Technical Design Report
TTC	Timing, Trigger and Control
TX	Transceiver
UC	Ultrascale
UC+	Ultrascale +
VLAN	Virtual Local Area Network
VME	Versa Module Europa
VS	Vertical Stabilization
VTRx+	Versatile TransReceiver Plus
W/R	Write and Read
XFEL	Xray Free Electron Laser
XVC	Xilinx Virtual Cable

SUMMARY

1 INTRODUCTION

The traditional Field Programmable Gate Array (FPGA) usage was addressed to be a platform for Application-Specific Integrated Circuit (ASIC) prototypes. Even nowadays, it is possible to find projects that apply the FPGA usage in this way (PATEL; SINGH, 2017). However, with the evolution of the FPGA families, it is common to see large applications developed over those devices, instead of the ASICs (HUDA; ANDRESON, 2017).

The newer FPGA families are able to handle several Gigabit per second (Gbps) input/output (I/O) channels implemented using high speed serial link protocols (BAUSS et al., 2016) (LEE et al., 2016) (LIU et al., 2016), operating in high internal clock frequencies (BUHROW; GOETZINGER; GILBERT, 2016) and also of serving as future downstream coprocessors due its internal DSPs resources (HILL, 2011).

A downstream system duty is to receive a specific data format from the upstream side. The data received should be processed, reformatted, labeled, filtered, routed according to the application. Those functionalities can be implemented and accelerated by ASICs, FPGAs, Digital Signal Processors (DSP), Graphics Processing Units (GPU), etc (ZHANG et al., 2011) (VALDERRAMA et al., 2011) (NGUYEN et al., 2017).

In the same way, an upstream readout system goal is to format data to send it to the downstream. The data to be transfered can be coded, converted, concentrated and formatted to be recognized by the receiver. Examples of upstream systems are sensors, Analog to Digital Converters (ADC's), concentrators, data bases, etc.

The evolution of FPGA Multi Gigabit Transceivers (MGT) brought opportunities for its application as large final products in several areas. Some examples are: power systems (NGUYEN et al., 2017), aerospace instrumentation (ALENA et al., 2016), climatology instrumentation (DEVAKUMAR et al., 2016), network computers (BUHROW; GOETZINGER; GILBERT, 2016), physics instrumentation (BAUSS et al., 2016), among others.

Downstream systems under development need an emulator to act as data source and to help the debugging. On other hand, the upstreams require a receiver to evaluate latency, data integrity and data format. FPGA systems are capable to emulate sourcing systems or to receive the data as a sink system, limited only by the hardware capability, such as number of I/O channels and transceivers maximum data rate (LLOYD; GOKHALE, 2016). The FPGA usage as emulators for real time testing enhances adaptability and decreases the time of the debugging (XIONG et al., 2016).

To have a means to test upstream or downstream projects, this work implemented the Data Sourcing System (DSS), that is composed by two FPGA Designs. The Data Sourcing (DS) Sender is able to produce and emulate any upstream data synchronized in different channels with total bandwidth of 320 Gbps per board. The DS Receiver is able to receive the same amount of data, to evaluate the latency and to check the link integrity.

The signal distortion increases with the serial data rate speed. Using link integrity measurements is possible to optimize MGT channels tuning parameters. The tuning parameters are used to compensate those impairments in the signal and to reduce the bit error rate (BER) (XILINX, 2012). Thus, DS Receiver is able to check the BER according to the tuning settings.

The DSS was implemented and tested using the Pulsar 2b board, a custom FPGA-based Advanced Telecommunication Computing Architecture (ATCA) board designed by Fermi National Accelerator Laboratory (FNAL) to be a scalable high speed link processor system (OLSEN; LIU; OKUMURA, 2014), which will be described in Section 1.2. The Pulsar 2b board has as core a Virtex 7 Xilinx FPGA with 40 of its MGTs connected to a Rear Transition Module (RTM). In this work, the focus will be the implementation and test of those RTM links.

The complete system has two ATCA shelves with 24 boards interconnected through quad small form-factor pluggable plus (QSFP+) optical fibers, which increases the number of channels to 480 and the total bandwidth up to roughly 3.8 Tbps. One shelf performs the DS Sender emulator, and the other performs the DS Receiver to record the data and to check latency, synchronization and bit error rate (BER). The management and debugging are performed using IPbus protocol.

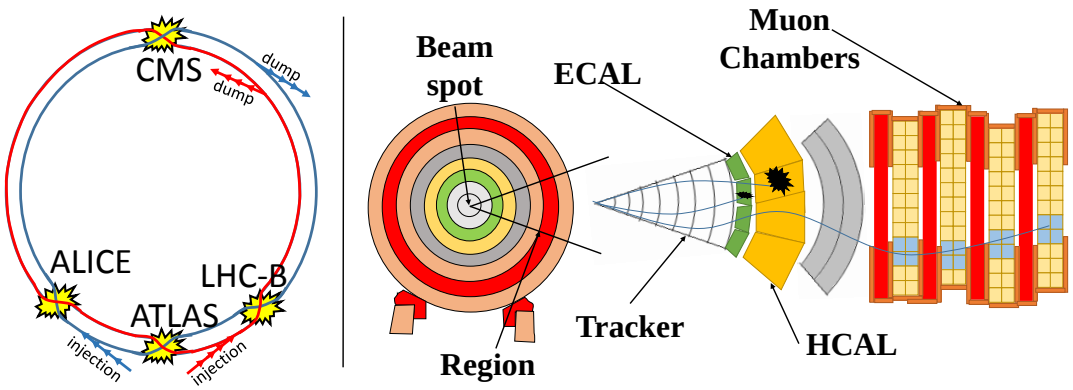
This thesis details the whole DSS implementation, performance evaluation and functionalities in board and shelf level scenarios. The results have shown that the DSS is a generic system that can be applied to test, debug and demonstrate any up or downstream system.

A case of study is also presented, describing the DSS usage to demonstrate the performance of a High Energy Physics (HEP) data formatter. The following subsection summarizes the relation between DSS in this HEP application.

1.1 LHC and CMS Context

The Large Hadron Collider (LHC) is a particle accelerator designed and/or built to provide proton-proton collisions for HEP experiments (TOMEI, 2012). The LHC machine is the largest and most sophisticated scientific instrument ever built, illustrated in the left of Figure 1. The accelerator and experiment detectors are led by the European Organization for Nuclear Research (CERN) and located near Geneva between Switzerland and France border at 100 meters beneath the surface.

Figure 1 - Left - LHC Machine. Right - CMS Subdetectors.



Source: Adapted from (TOMEI, 2012).

This proton accelerator has 27 km of main ring perimeter and 4 collision points. The protons are grouped in bunches and accelerated in two beams in opposite directions in the ring. The bunch crossing (BX) happens in the 4 collision points every 25 nanoseconds (40 MHz), where multiple collisions between protons can occur simultaneously. Each one of the collision points has a diferent experiment detector to monitor the collisions. There are four experiment detectors: A Toroidal LHC ApparatuS (ATLAS); Compact Muon Solenoid (CMS); A Large Ion Collider (Alice); and LHC beauty (LHCb) (TOMEI, 2012).

The CMS (CMS COLLABORATION, 2008) detector was designed to register, directly or indirectly, the track and energy of all the Standard Model particles (TOMEI, 2012) that are resultant of the collisions. It is made of several layers of specialized sub-detectors to measure the different characteristics of the produced particles, as illustrated jn the right Figure 1. The CMS instrumentation includes different components: Muon Chamber, Hadronic Calorimeter (HCAL), Eletromagnetic Calorimeter (ECAL) and the Silicon Tracker. Therefore there is a superconducting solenoid that generates a high magnetic field capable of bending the charged particles resultant from the collisions. Thus, as higher is the particle energy smaller is the effect that its track will receive from the magnetic field.

Through the analysis of the track and energy, detected and measured by the tracker, muon chamber and the calorimeters, it is possible to reconstruct events resultants from the collisions. Due to the huge amount of events and collisions, the data generated by the different types of sub-detector should pass through an online filter system called Level 1 (L1) Trigger. The L1 trigger performs a real time low resolution reconstruction to select relevant physics events. The main goal of the L1 trigger is to save bandwidth and storage resources. A better and more detailed description of the CMS global trigger is presented in (TAUROK; BERGAUER; PADRTA, 2001) (ACOSTA, 2016). The Silicon Tracker is the only layer that does not take part of the CMS L1 trigger system in the current LHC run, known as LHC Run 2.

The LHC Run 3 upgrade (KLEIN, 2017) will increase the current 140 collisions per bunch

crossing for 200 collisions (CMS COLLABORATION, 2015) (ACOSTA, 2016). This fact poses unique challenges for the L1 trigger requirements in algorithms and architecture (CMS COLLABORATION, 2015) (BROOKE, 2012). In the CMS Phase 2 upgrade the L1 trigger system will include information from the Outer Silicon Tracker, which will help CMS maintain a rich and efficient physics program during this period. This new part of the filter system is known as L1 Tracking Trigger (L1TT). A silicon-based tracker online trigger has never been realized.

The CMS L1TT will require an online reconstruction of charged particle tracks for every bunch crossing at 40 MHz, from a huge amount of input data. This scenario will need a high speed communication and massive pattern recognition capability, to perform the selection of tracks from high energy particles, greater than 2 Giga Eletrovolt (GeV), in a maximum latency of 4 μs (ACOSTA, 2016) (CMS COLLABORATION, 2015).

This work is part of a international collaboration, led by Fermilab, that proposed with the contribuition of the system described in this thesis, a Level 1 trigger for the CMS Outer Tracker. The DSS, system described by this thesis, was firstly idealized to be the emulator of CMS Outer Tracker. The goal of the DSS in the collaboration was to aid the test, debug and demonstration of the proposed L1 Tracking Trigger downstream system. More details of this case of study are described in Chapter 6.

The complete hardware demonstration were performed at FNAL. The hardware used for the DSS development and the proposed trigger testing is summarized in Section 1.2.

1.2 Hardware Summary

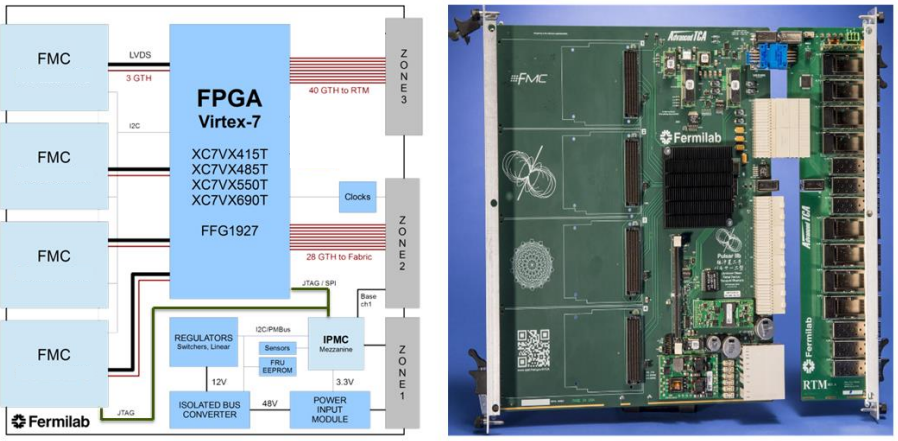
The Fermi National Accelerator Laboratory (FNAL) Research and Development (R&D) team designed the Pulsar 2b board, illustrated in Figure 2, for applications that require a full-mesh ATCA backplane to share data inside the ATCA shelf.

Moreover, the board has massive I/O capabilities through RTM boards with a maximum of 400 Gbps (OLSEN; LIU; OKUMURA, 2014). The main components of the board are listed below:

Xilinx Virtex 7 FPGA Pin compatible with many part numbers. However, the part number used in this work was the XC7VX690T-FFG1927-2. This FPGA has 80 MGTs available. Xilinx 7 series FPGAs MGTs are called as GTH transceivers (XILINX, 2015b). Details about GTH distribution are described in Section 3.2.

FMC Mezzanines Four FPGA Mezzanine Card (FMC) slots are available to insert mezzanines for specific applications. Pulsar has 3 MGTs per mezzanine.

Figure 2 - Left: Pulsar 2b block diagram. Right: Photo of Pulsar 2b board with RTM.



Source: Adapted from (OLSEN; LIU; OKUMURA, 2014).

Zone Connectors The backplane interfaces include power, high speed for Gigabit links, clock distribuition and the Intelligent Platform Management Interface (IPMI) communication. More details in Section 3.1.1.

RTM MGTs 40 MGTs to QSFP+ connectors for cooper or optical fiber cables.

Backplane 26 MGTs, 2 per each of the other 13 boards inside the ATCA shelf.

IPMC Mezzanine Card The Intelligent Platform Management Controller (IPMC) is implemented in a microcontroller. It is required by ATCA boards for monitoring and controlling through the IPMI protocol. It also provides Transmission Control Protocol over Internet Protocol (TCP/IP) services for user remote control.

The DSS applied in one board is able to transfer or receive data at 8 Gbps per RTM channel. Thus, the total bandwidth per board is 40*8= 320 Gbps. An ATCA shelf composed by 12 boards operates at 12*40*8= 3.840 Tbps.

1.3 Goals and Contribuitions

The work described by this thesis aimed to:

- Describe the Data Sourcing System implementation: The DSS system was implemented for Pulsar 2b Virtex 7 FPGA. The development goal is to provide a testing system on FPGA design that allows user remote controlling, syncronization and features for metric measurements, as latency, BER, among others. The thesis should allow the reproduction of this system for other Printed Circuit Boards (PCB) and FPGAs.

- Implement link latency measurements: The DSS is able to measure the link latency that consider the serial link protocol processing in the transmitter (TX) and receiver (RX) sides plus the signal propagation in the media.
- Implement BER measurements: The link integrity measurement is desirable to characterize the data reliability for any up or downstream system under development.
- Implement Remote Tuning: With the increase of the speed in the links, a remote tuning feature can help to optimize tuning parameters and avoid bit errors in the reception.
- Present an ATCA shelf level hardware evaluation: The FNAL hardware setup was implemented with full optical cable interconnection between ATCA shelves. The DSS is able to evaluate latency, syncronization and BER of a 3.8 Tbps system.
- Describe a case of study of a downstream testing: As case of study, the DSS was used by HEP application to emulate CMS fron-end (FE) electronics and to test the back-end (BE) downstream system. The system helped the development and demonstration of a proposed trigger for the CMS experiment.

The contributions are:

- User remote controlling: The DSS is fully remote controllable through an user interface. These feature avoids that several Joint Test Action Group (JTAG) pins are connected in the debug process, which brings a scalability of the system when several boards and links are part of the setup.
- Evaluation of a 3.8 Tbps setup: An FPGA-based setup was never characterized with such a data bandwidth. BER, latency and synchronization measurements were performed separately for each one of the 480 channels in the system.
- Evaluation of tuning optimization process: The DSS is able to perform remote tuning and evaluate the effects that a parameter optimization of the channels would improve the link integrity performance.
- Proposed L1TT demonstration: The DSS is part of an international collaboration that demonstrated a proposal for the CMS trigger system. The downstream first step was developed by using DSS as a CMS Outer Tracker emulator with the same data amount, speed and latency required by the CMS management board.

1.4 Document Organization

This document is organized as follows. After the introduction and context presented in Chapter 1, Chapter 2 describes some related work. The whole hardware definition and details

about the ATCA setup implemented at Fermilab are described in Chapter 3. Chapter 4 details the FPGA firmwares of the DS Sender and Receiver. The tests and results of the DSS applied in a generic way in board and shelf level scenario are presented in Chapter 5. Chapter 6 details the case study and L1TT demonstration results. Finaly, the conclusions, future work and publications achieved are listed in Chapter 7.

2 RELATED WORK

Several areas have been using FPGA devices for the development and testing of upstream and downstream systems. The newer FPGA families presents greater number of I/Os, flip-flops, internal DSPs, block Random Access Memories (RAMs), among other resources, when compared to the older families. This FPGAs hardware resources evolution can be noticed in Table 1 among Xilinx Virtex FPGA families 5, 6, 7, Ultrascale (UC) and Ultrascale plus (UC+). The table shows the maximum capability considering all Virtex part numbers of each family.

Table 1 - Xilinx FPGA Families Evolution.

Xilinx FPGAs	FPGA Resources					
	Max Number of Flip-Flops	Max Number of DSPs	Max Number of Block RAMs (Mb)	Max Number of MGTs	Max Total MGT Bandwidth	Max User I/Os
Virtex 5	207 K	1,056	16.4	48	600 Gbps	1,200
Virtex 6	948 K	2,016	38.3	72	914.4 Gbps	1,200
Virtex 7	2,443 K	3,600	67.7	96	1.392 Tbps	1,200
Virtex UC	5,065 K	5,520	132.9	120	2.808 Tbps	1,456
Virtex UC+	3,456 K	12,288	94.5	128	4.192 Tbps	832

Source: (XILINX, 2015d), (XILINX, 2015e), (XILINX, 2017a), (XILINX, 2016b), (XILINX, 2017b).

The development of the upstream and downstream systems should consider also the channel performance. The application requirements can be related to bandwidth, latency, link integrity, synchronism, among other metrics.

To have a better organization of the different applications and projects, this thesis classifies the hardware setups between the requirement reached and labeling them as up and downstream developments. Therefore, the publications are organized in sections as follows. The Section 2.1 lists the publications of developments that have the bandwidth and latency as focus. In the same way the requirements of synchronization and link integrity are listed at the Sections 2.2 and 2.3, respectively.

2.1 Bandwidth and Latency

The bandwidth and latency are requirements correlated between each other. As an FPGA has the capability to receive data input at high speed, the applications often need to process and to delivery them in a short time. This relation brings the idea of real timing requirements, which justifies the FPGA usage in those applications.

The real time requirements for digital data processing (downstream) are discussed at the paper (MAKOWSKI et al., 2015). The authors implemented a data processor in a Micro-TCA FPGA-based board. The bandwidth capability of 12.5 Gbps and link latency of 100 ns are used by Xray Free Electron Laser (XFEL) synchrotron experiment to control the Low Level Radio Frequency system.

Regarding applications at synchrotron light sources, it is possible to mention the Lawrence Berkeley National Laboratory (LBNL), where a prototype ATCA readout (upstream) system was built to capture X-ray data with a 1M pixel frame store charge-coupled device (CCD) sensor camera head at the LBNL Advanced Light Source (MCVITTIE et al., 2012). This camera is capable of generating image data at over 400 Mega bits per second (Mbps), which must be processed in real time and stored in disk. The processing system has an ATCA FPGA-based processor blade running the user interface software like perfoming image descrambling and formatting (downstream).

The work described in (ALENA et al., 2016), proposes an image processor with GPU, FPGA and RapidIO(RIO) interface to provide multiple Gbps of data communication. This image processor is a downstream system that receives images and spectra from multiple instruments and processes the data in real time to allow autonomous navigation for the National Aeronautics and Space Administration (NASA) spacecraft.

Another image processor is presented at (ZHANG et al., 2011). The stereo matching application is applied for object detection in computer vision. The papers in this area are used to proposed downstream systems using ASIC, GPU or FPGA platforms to achieve real time performance. The authors cited implemented an FPGA-based stereo matching with a performance of 60 frames per second in high definition.

The real time capability is also explored by the work in (DEVAKUMAR et al., 2016). This paper describes a (downstream) DAQ real time processor implemented in FPGA that processes lightning data. The Cloud to Ground Lightning flash has its prominent peak within the frequency range 3-30 KHz. The authors used an ADC sensor (upstream) and the time stamp information to determine the rise and fall times, pulse width, time domain signature, etc. The DAQ and smart triggering form an integral part of the FPGA implementation. The smart triggering logic implemented on the FPGA filters the valid lightning events and sends the lightning parameters to a central processing system.

An ADC upstream system was proposed by (FERENCI; BERROTH, 2015). The described system is able to sample 60 Gbps through 12 channels. According to the authors, the digital system developed can scale to more than 100 Gbps depending on the FPGA MGT evolution.

The electrical power and energy systems have been using FPGAs as cost reduction in real time applications. The consumer load monitoring in the domestic environment needs to be monetarily feasible to increase the popularity. Considering the requirements of hundreds of nanoseconds latency, FPGA is a good solution for smart meters (downstream) implementation, as described at (NGUYEN et al., 2017).

A greater bandwidth requirement can be found in other applications. (BUHROW; GOETZINGER; GILBERT, 2016) presented a digital downstream system implemented for a Virtex Ultrascale FPGA that can support IP tunnels at 1 Tbps. Thus, the digital system in network computers for this data rate demands operation at 400 MHz. The investigated application tries to use the FPGA to hold IP secure communication (IP-Sec) protocol and increase the speed. To do so, the authors implemented a system that can work with wide buses of 1024 bits to reduce the segmentation of the IP packets.

Due to the number of I/Os presented in the FPGA devices, it is common to find aplications that employ them as data delivery, data formatting or data processing system in the downstream. One of these applications is in the physics area, which uses FPGA-based platform for trigger systems. An example can be found at (BAUSS et al., 2016). The ATLAS Calorimeter uses 4 Virtex UltraScale FPGAs as trigger system dedicated to process the jet event data. This event is used to produce 2 Tbps of data to be analyzed in hardware in a few hundreds of nanoseconds latency. The maximum capability of the board proposed is 3.6 Tbps input bandwidth.

2.2 Synchronization

The ATCA standard is used to propagate signals between systems that share the same shelf. (BATISTA et al., 2009) describes the Vertical Stabilization (VS) (downstream) system, which has to control the stability of the plasma in the Joint European Torus (JET) Tokamak fusion experiment. One of the most important project requirements is the control of the closed loop-cycle within 50 μs with a maximum jitter of 2.5 μs. Two communications fabrics are used simultaneously on the ATCA backplane, the Peripheral Component Interconnect Express (PCIe) and the Aurora 8b10b 2.5 Gbps of Line Rate in star and full-mesh topologies respectively. The Aurora protocol is also used in the optical links. The common clock and some other signals are sent through the backplane to synchronize all the electronics.

Even when the system presents electronics that are not under the same communication backplane, the signals can also be propagated by the fiber optic cable. The article in (LEMKE et al., 2009) describes a part of the DAQ system as downstream for the Compressed Baryonic

Matter (CBM) experiment. The authors say that the most important ability to enable synchronization over fiber links is to guarantee a deterministic latency over them. Therefore the FPGAs serial transceivers have to be configured in a special way for always starting with the same basic latency after link initialization. The signals synchronization, clock distribution, control messages and data streams are spread over the electronics through a single bidirectional fiber link. The link is used to synchronize the upstream front-end electronics and the downstream DAQ. The time syncronization messages carries a deterministic latency of the system and information of the next restart.

The fiber optic cables can also provide a synchronization network between telecommunication crates, as Versa Module Europa (VME), ATCA, etc. In the work of the authors (BELLATO et al., 2013) and (BARRIENTOS et al., 2012), the ATCA carrier board of the Advanced GAmma Tracking Array (AGATA) receives the optical synchronization signal through a mezzanine connected with an optical cable to a clock source. The common clock and time stamps signals control the trigger downstream system of the experiment.

A similar synchronization scheme is applied at the LHC on some of its experiments like CMS (TAUROK; BERGAUER; PADRTA, 2001) (BUNKOWSKI et al., 2007) and ATLAS (HIDVEGI, 2011). The LHC machine has a beam collision frequency of 40 MHz and the trigger system should identify interesting physics events resultant from each collision. Every time the trigger identifies a good event, the L1 Accept (L1A) signal is sent through the Timing, Trigger and Control (TTC) optical synchronization network. The TTC electronics are implemented in VME Trigger Control System (TCS). This network is composed by the readout front end electronics (upstream system) and the L1 trigger and the DAQ (downstream) system. The L1A signal together with the current collision tag, called bunch crossing (BX) number, are sent to the readout electronics with a common clock domain. Then the readout sends the data to the DAQ (downstream system). The TTC network also should reset all electronics every LHC orbit that is composed by 3564 BX. The signal responsible for that reset is called bunch crossing zero (BC0). This trigger system is still used at LHC.

The authors of (MARIN et al., 2015) investigated the latency and clock synchronization schemes for different MGT families. All MGT Intellectual Property cores (IP cores) studied have a default and a low-latency versions. They can present deterministic or stochastic latency. The authors emphasize that, considering the TTC applications where the synchronization is critical, the MGT choosen should have deterministic latency.

The determistic latency is achieved using channel bonding. The authors of (SARMAH; AZEEMUDDIN, 2015) describe how channel bonding schemes can allow syncronization between channels. Those schemes can synchronize non-deterministic latency encoding, like in the MGT IP core Xilinx 64b/66b, through First-In-First-Out (FIFO) based channel bonding implementations. Despite it slightly increases the application latency, it ensures a better syn-

chronization without reducing performance of bandwidth and link integrity (BER) metrics. In exchange more FPGA resources are used.

2.3 Link Integrity

The signal distortion increases with the serial data rate. Taking into account high speed signaling, the bit sampling in the receiver side has a smaller time window to happen. Moreover, the inter-symbol interference (ISI) also increases with the speed and can effect the sampling of neighboring symbols (XILINX, 2012).

Those impairments effects decrease the link integrity and cause bit errors in the data. There are many ways to tackle this problem. The wrapper encoding protocols apply techniques to identify or even correct the data in the reception. The polar codes, investigated by (XIONG et al., 2016), need testing and measurements of minimum 10^{13} bits to ensure reliability. Thus, the authors propose to use an FPGA data emulator to reduce the time consuming debug process. This reduction is possible due to the high speed serial links of the MGTs.

Another test scheme can be found at the thesis (ZOU, 2017). The author uses FPGA emulation to evaluate the link integrity and the performance of low density parity check. The architecture implemented in an FPGA uses a pseudorandom binary sequence (PRBS) generator for the input in the encoding. The link errors were emulated using a gaussian error generator.

The channel equalization is also one method to enhace the link integrity. (ALOISO et al., 2016) describes an auto-adaptive serial link that is able to find optimal RX clock and tune parameters based in BER requirements. The system used two Xilinx Kintex 7 Evaluation boards, 8b/10b encoding and the line rate up to 10.3125 Gbps.

Similar systems can be found for specific applications, like in (DAMLE et al., 2015) that uses BER and jitter to characterize a high speed serial link. (FRANS et al., 2015) describe another adaptative system that can change the tuning parameters and speed rate to achieve 10^{-15} of BER.

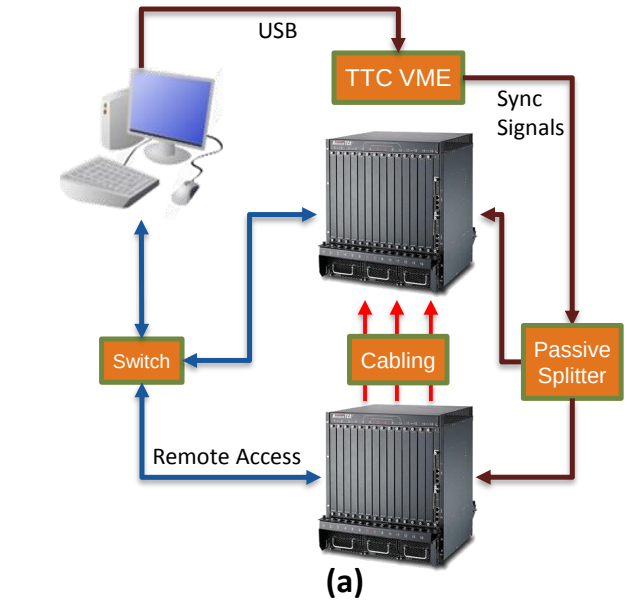
2.4 Summary

A system with a large bandwidth capability is described by this thesis. The DSS is able to provide/ receive data in hundreds of nanoseconds latency and with the total bandwidth of 3.8 Tbps. It uses part of TTC system as synchronization signals source for the hardware setup. The DSS isolates the MGT non-deterministic latency with a FIFO scheme and the TTC signals ensures the data arrival in a small time window. A tuning optimization was performed and is described in Subsection 4.4.1. The Chapter 3 describes the hardware setup used by this work.

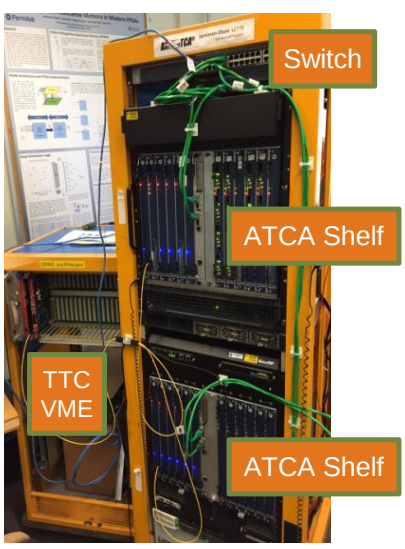
3 HARDWARE DEFINITION

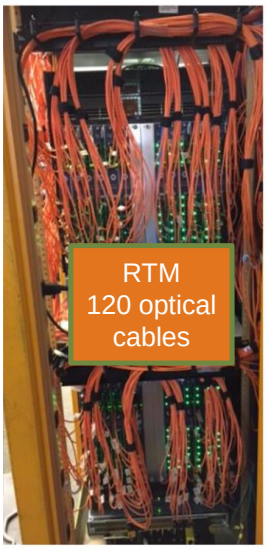
The hardware setup used by this work was assembled at FNAL and it is illustrated in Figure 3. The electronic testbench started with two ATCA shelves, a VME crate and a Linux computer used as gateway to the ATCA network.

Figure 3 - (a) ATCA Infrastructure Elements. (b) ATCA shelves Front Side. (c) ATCA shelves Rear Side.



(a)





(b) (c)

Source: Created by the author.

As can be seen in Figure 3(a), there is a computer used as gateway for the ATCA network. The computer provides remote communication for all Pulsar 2b IPMCs and FPGAs in both ATCA shelves.

Through this network, it is possible to use the Xilinx Virtual Cable (XVC) protocol to remotely program and debug the FPGA boards (RAMALHO et al., 2015). Moreover, it is possible to access FPGA TCP/IP services like IPbus (see subsection 4.2) and perform write/read operations in the FPGA memory blocks.

Each ATCA shelf has 12 Pulsar 2b boards installed and interconnected in the rear side by the RTM optical cabling, illustrated in Figure 3(c). Tests were performed to evaluate those optical links performance.

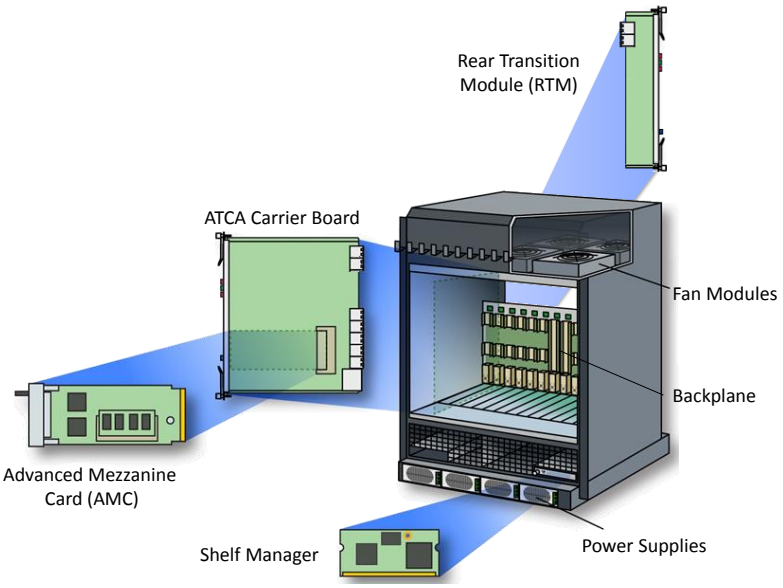
The optical cables used are from 3 different vendors and have length between 3-6 meters.

3.1 ATCA Concepts

The ATCA is a standard developed to provide high-speed communication interfaces for telecommunication, military and aerospace industries, being also used in HEP instrumentation (PEREK; MAKOWSKI, 2011) (LARSEN, 2012). In this architecture, a shelf creates a controlled environment for user specified boards to operate with optimal performance, monitoring the temperature, controlling fans and managing the power supply, for example.

The main elements of an ATCA shelf are illustrated in Figure 4.

Figure 4 - ATCA elements.



Source: Adapted from (PICMG, 2008).

The ATCA standard defines the backplane high speed connection to be used by carrier

boards. The backplane and physical slots for board connection are provided by an element called ATCA shelf and the boards compatible to the physical slots (physlots) are known as ATCA carrier boards. Through the physical slot backplane, one board can have direct electrical serial link with the other slots. In general, the backplane has a dedicated physical slot for switch/hub carrier boards. The carrier boards, like the name suggests, can carry mezzanine boards that have custom electronics and functions. The ATCA standard also allows each physical slot to have a rear module called RTM, as seen on Figure 4. There are a few standards for mezzanines used in ATCA, such as Advanced Mezzanine Cards (AMC) or FMC. They define the characteristics of the boards, such as dimensions and connectors.

The ATCA carrier boards must have support to communicate with the Shelf Manager card, which is responsible for the management of all the carrier boards. That communication allows Shelf Manager to monitor the sensors present in each board installed in the shelf and control the system operation, protecting the boards from damages and ensuring its best performance. The AMC sensors can be monitored in the same way. For example, if a temperature sensor indicates an overheating in a board, the Shelf Manager will communicate with the fan modules and increase their speed. The power supplies resources are also managed by Shelf Manager. This ATCA management and control concept is called Hardware Platform Management (HPM) (PEREK; MAKOWSKI, 2011) (PAIVA, 2016).

3.1.1 ATCA Backplane Zone Connectors

The ATCA shelf is composed by several ATCA carrier boards connected between them by a backplane. This feature presents several advantages concerning to the usual connections operated by cables, like easily plugging and unpluging boards and the prevention of errors due to connecting cables to the wrong connectors. Also, considering that the quality of high-frequency signals is influenced by temperature, the system is more reliable due to the decrease of temperature by the backplane usage instead of cables (PEREK; MAKOWSKI, 2011).

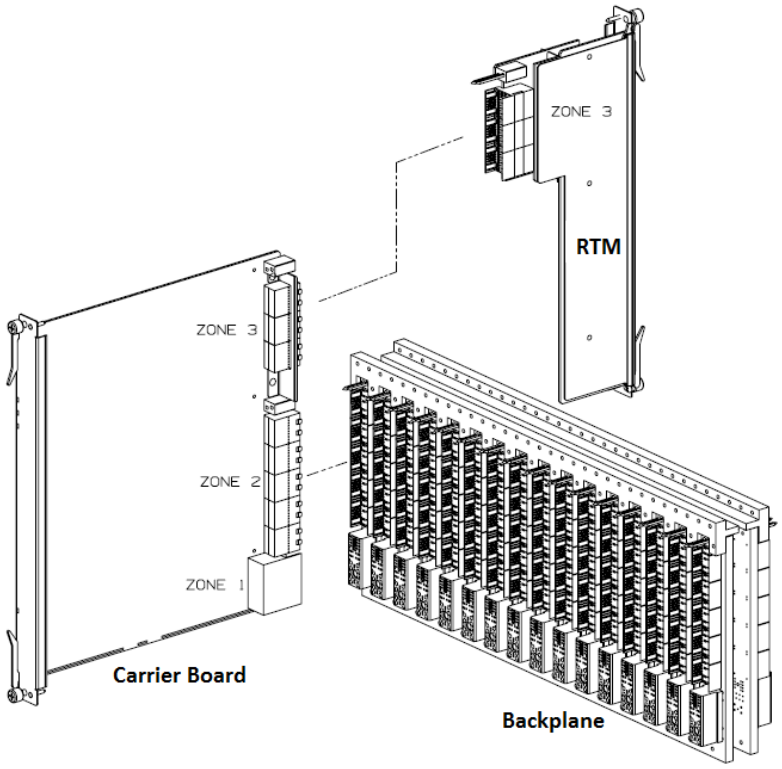
The ATCA backplane defines three zones of connections for each physical slot, as illustrated in Figure 5.

The Zone 1 connector is used mainly for power supply and management through the IPMI protocol. It contains the metallic test bus, the ringing generator bus, and low-level HPM signals.

The base interface, fabric interface, update channel interface, and synchronization clock interface signals are distributed through the Zone 2 connector.

The base interface is a dual star topology with the switch/hub slots being able to communicate with all the other boards. Base channels are comprised of four differential signal, two for TX and two for RX, capable of supporting 10/100/1000 BASE-T signaling. Base switch/Hub slots have base channel connections to up to 13 other physical slots. Switch slots provide also

Figure 5 - Backplane Zones connections.



Source: Adapted from (PICMG, 2008).

connections to the Shelf Manager (PICMG, 2008). Its purpose is usually to enable the user to communicate with each carrier board, acting as an ATCA network gateway.

The fabric interface is used for communication between the carrier boards, and is capable of supporting a number of topologies. Backplanes can be designed for a specific topology, such as dual star or full-mesh (where all boards communicate with all the others). All these fabric interface implementations are well defined on the PCI Industrial Computer Manufacturers Group (PICMG) specification version 3.0 (PICMG, 2008).

Those network topologies present at base and fabric interfaces are in general externalized by an ATCA switch connected in the shelf. Although both interfaces have connections with the same electronic device (carrier board), they have a physical separation between themselves in the backplane.

The update channel interface is comprised of 10 differential signal pairs in a point to point connection between two slots. Typically, the update channel is used between adjacent slots in a backplane; however, it can be routed between any two slots. For example, non-adjacent slot routing can be used to support double-wide boards (a board that occupies two physical slots at the same time) (PICMG, 2008).

The synchronization clock interface provides a set of clock buses to enable applications

that require the exchange of synchronous timing information among multiple boards in a shelf (PICMG, 2008).

Using the Zone 3 connector, the carrier board can connect to a Rear Transition Module (RTM). These devices can extend the board functionalities, like transfering data directly from the carrier board to non-ATCA devices or other ATCA shelves. The RTM can perform hotswap operations in a Module Management Controller (MMC), that is present and connected to the IPMC through an Intelligent Platform Management Bus - Local (IPMB-L) (PAIVA, 2016).

3.1.2 ATCA Carrier Board

The ATCA carrier board should have a device called IPMC. This device must support IPMI communication and is responsible for monitoring the carrier board sensors and sending their data to the Shelf Manager using the Zone 1 HPM links. If an AMC is connected to the carrier board, it must have a Module Management Controller (MMC). These are devices that operate the communication between the AMC MMC and the carrier board IPMC using the IPMI protocol. The purpose of the MMC to the AMC is similar to the IPMC duty to the carrier board, although the MMC is usually simpler and less resourceful than the IPMC. More details about HPM and IPMI can be found in (PAIVA, 2016).

The carrier board must be able to be plugged and unplugged without the need turning the ATCA shelf off. This sort of operation is called "hotswap", and is performed by the IPMC when a handle switch in the board is activated. For example, when a module is inserted in the shelf, the power management provides only enough power for the IPMC be turned on. After that, the Shelf Manager initiates the activation process that negotiates the power supply, sets the connections and reads information about the board. These steps turn the device in its full operation capability. Finally, more power is provided so that the whole board can be turned on. For the extraction of the module, the reverse process is performed. Any ATCA electronic is capable of performing hotswap operation is also known as Field Replaceable Unit (FRU).

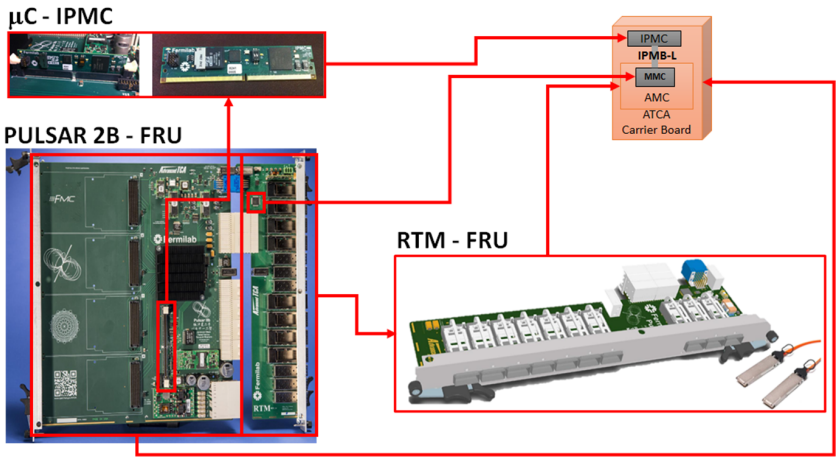
Therefore, to support hotswap the IPMC must be able to control the payload power inside the board in at least two modes. First, when is inactivated, just the IPMC receives power. Second, when it is activated, which represents full power and resources utilization of that board.

The ATCA carrier boards make use of Zone 2 to connect at base interface and fabric interfaces of the shelf to communicate with the other carrier boards.

3.2 ATCA Hardware Setup

The interconnections between the ATCA elements and the hardware setup in this work used is illustrated in Figure 6.

Figure 6 - ATCA elements applied to the hardware setup used.



Source: Created by the author.

The ATCA shelves are composed by two different ATCA carrier boards: an ATCA switch blade to externalize the network connection and the Pulsar 2b board.

The Pulsar 2b is capable of carrying up to four mezzanine cards to implement specific functionalities, extending the board capabilities. Each mezzanine slot has 3 GTH [1] to provide a high speed communication channel between mezzanines and the FPGA. They also have several LVDS connections to the Virtex 7 to tranfer data, clock and reset signals.

In the DSS, the wrapper is responsible to manage the 40 RTM available channels. Those channels are connected to 40 GTH transceivers/receivers on the Pulsar 2b Virtex 7 FPGA (34 from left bank in the FPGA and 6 from right), as illustrated in Figure 7.

The GTH mapping, on Figure 7, is described as follows. The X# is related with the FPGA side where a GTH is placed (0 means left, 1 means right). The Y# is an index from bottom (0) to up (39) for each GTH placed [2].

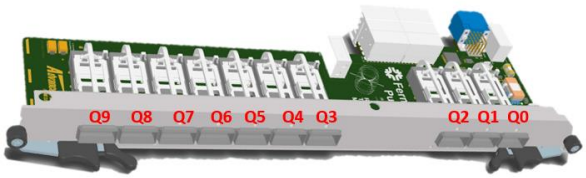
The ATCA shelves have 14 physical slots. The boards disposition in the ATCA shelves are illustrated in Figure 8. The Pulsar 2b boards are inserted at physical slots (physlots) 1 to 6 and 9 to 14. The physlots 7 and 8 are reserved for ATCA hub/switch boards.

An FMC slot is used at one board per shelf to install the TTC FMC mezzanine. More details of the TTC hardware is described on Section 3.3.

[1] Xilinx 7 series MGTs are called GTH transceivers

[2] The Virtex 7 placed at Pulsar 2b has 80 GTH available (40 in the left bank of the FPGA and 40 in the right)

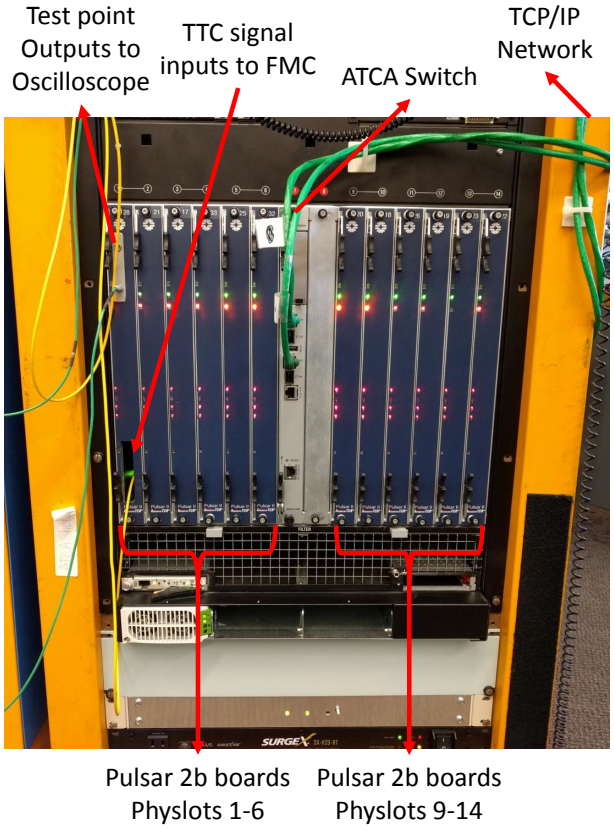
Figure 7 - RTM GTH Relation.



	GTH assigment in each RTM connector			
Q0	X0Y30	X0Y31	X0Y32	X0Y33
Q1	X0Y26	X0Y27	X0Y28	X0Y29
Q2	X0Y22	X0Y23	X0Y24	X0Y25
Q3	X0Y18	X0Y19	X0Y20	X0Y21
Q4	X0Y14	X0Y15	X0Y16	X0Y17
Q5	X0Y10	X0Y11	X0Y12	X0Y13
Q6	X0Y6	X0Y7	X0Y8	X0Y9
Q7	X0Y2	X0Y3	X0Y4	X0Y5
Q8	X1Y1	X1Y0	X0Y0	X0Y1
Q9	X1Y5	X1Y4	X1Y3	X1Y2

Source: Created by the author.

Figure 8 - ATCA shelf front view.

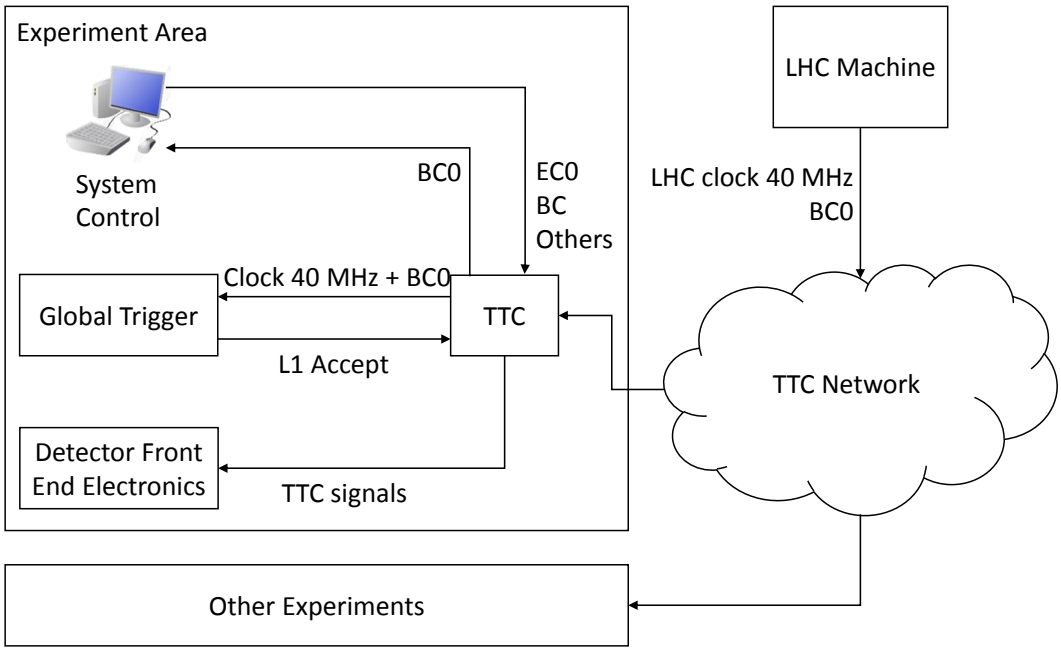


Source: Created by the author.

3.3 TTC Hardware

The TTC signals are delivered to several destinations over a passive optical fiber network (TAYLOR, 1998). This TTC network has several layers to distribute the signals from the LHC machine, to the front-end and back-end electronics of the detectors, as illustrated in Figure 9.

Figure 9 - TTC signal network.



Source: Adapted from (TAYLOR, 1998).

The TTC signals presented in Figure 9 are:

LHC Clock 40 MHz Bunch crossing frequency.

BC0 - LHC Orbit 11 KHz Bunch crossing zero tags the LHC orbit frequency, which is equivalent to 3654 bunch crossings.

BC BX Counter identification number.

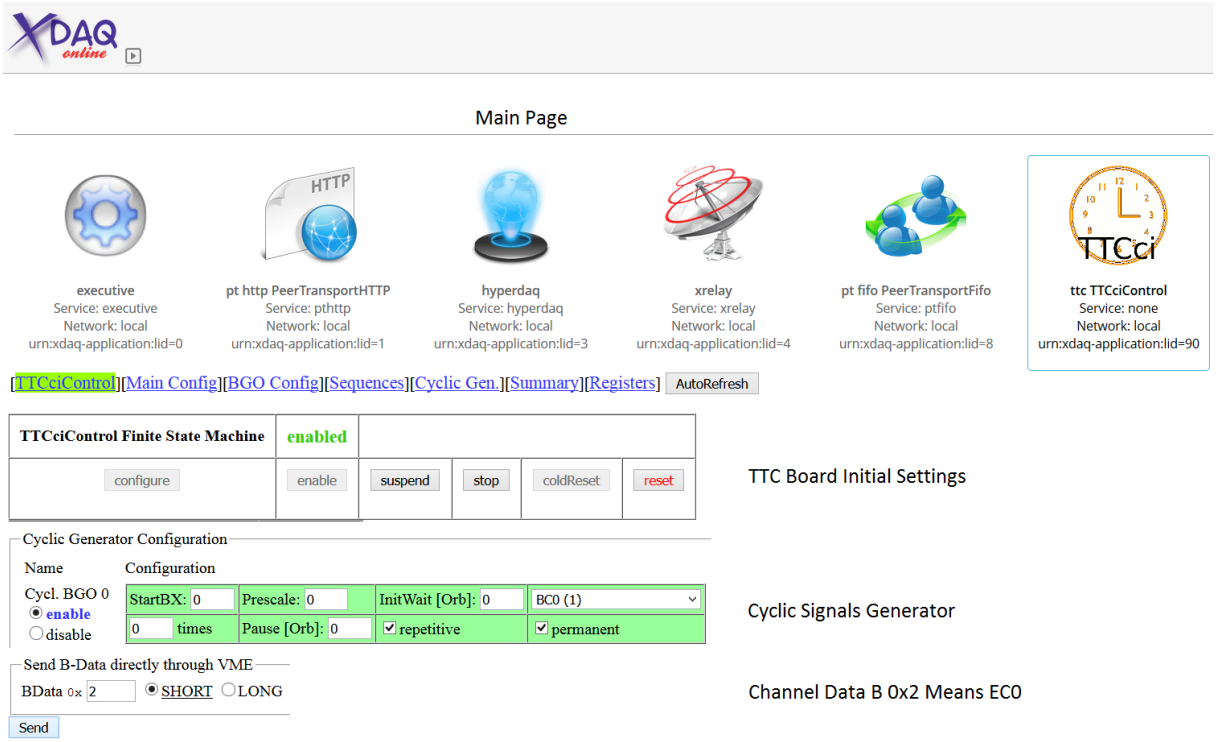
L1 Accept Flag of trigger acceptance.

EC0 Event counter reset.

This synchronism implemented at TTC system was developed by CERN over VME hardware (INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS - IEEE, 1987) (TAUROK; BERGAUER; PADRTA, 2001). The hardware setup used in this work has a VME crate with a TTC board to emulate and reproduce the required synchronism for the ATCA shelves.

The TTC hardware receives a USB connection from the computer in a VME board named CAENUSB, which is responsible for providing an interface to the other boards in the VME crate. Thus, the user configurations pass through CAENUSB to TTCci VME board. The TTC user settings are performed by the computer through the XDAQ system (CMS COLLABORATION, 2009). The XDAQ interface is illustrated in Figure 10.

Figure 10 - XDAQ System Web interface.



Source: Created by the author.

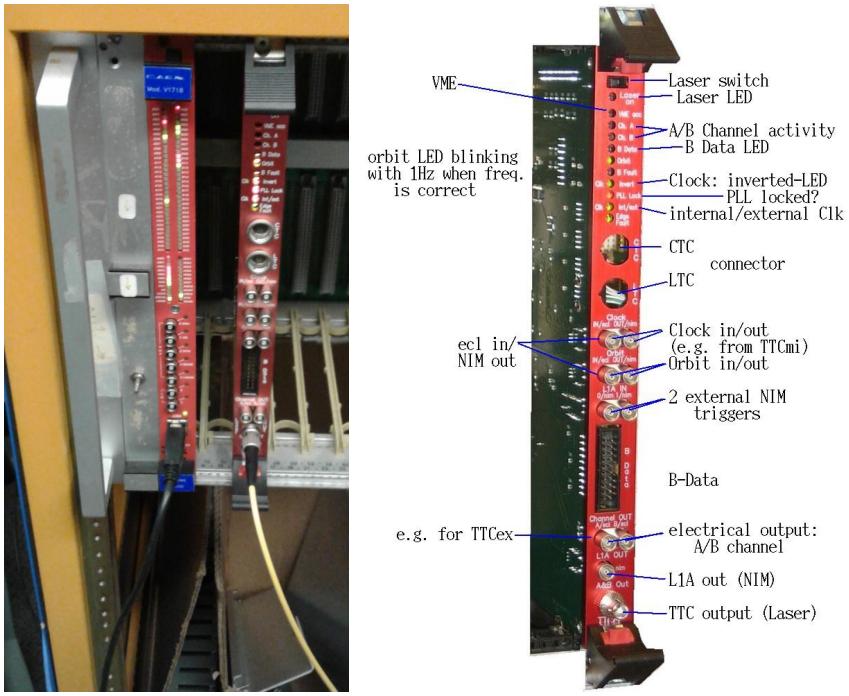
The TTCci board can receive signals from an external source, i.e. from the TTC network upper layers. Another optional functionality, which is used by this work, is to generate the TTC signals in the TTCci board according to user configurations.

Figure 11(a) is a photo of the VME crate used in the FNAL hardware setup. The TTCci front painel is illustrated in Figure 11(b).

The TTC output conector, shown in Figure 11(a), externalizes the signals to be used by the ATCA hardware. The optical signal passes through a passive splitter and is provided for both ATCA shelves. Each shelf receives the TTC signals in a Pulsar 2b in physical slot 1. The Pulsar 2b board in this case has the TTC FMC mezzanine card (OHWR, 2011) attached to it, as illustrated in Figure 12.

The board that has the TTC FMC needs to be set as the TTC clock master in the shelf. The Pulsar 2b IPMC microcontroller can be remotely accessed to enable that TTC signals pass through the Virtex FPGA and goes to the ATCA backplane. Only this board in the shelf should broadcast the signals in the ATCA.

Figure 11 - (a) TTC crate with boards. (b) TTCci front painel.



(a) (b)

Source: (a) Created by the author. (b) Adapted from (HOLZNER; CMS, 2007).

After the signals arrive in the ATCA shelves, they are processed and used by the Pulsar 2b FPGA. This processing is detailed in Section 4.1.

Figure 12 - TTC FMC attached to the Pulsar 2b board.

Source: Created by the author.

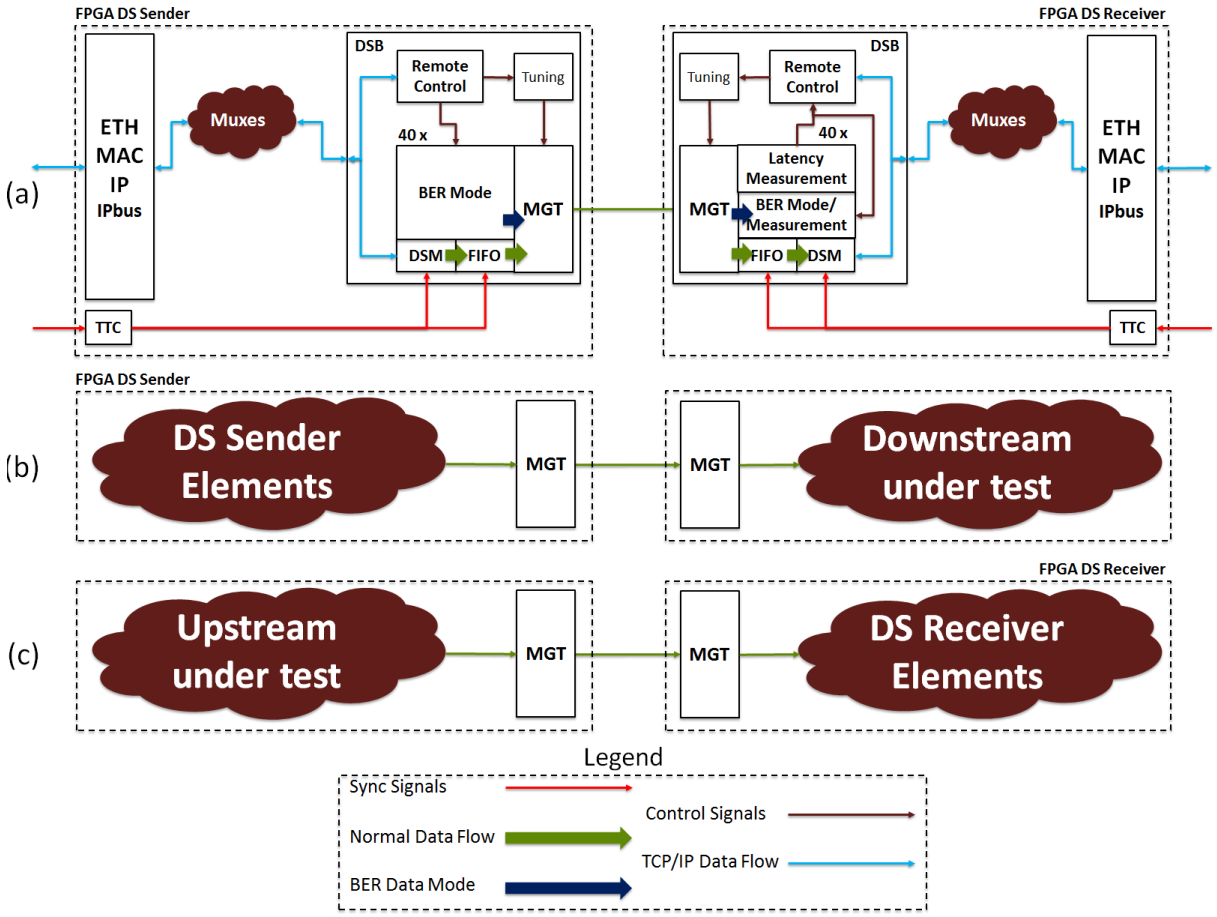
3.4 Chapter Summary

This Chapter presented the details about the hardware setup used. This setup is implemented at Fermilab and was used for the FPGA firmware developments that are described by Chapter 4. The results obtain by Chapters 5 and 6 also used the same setup.

4 THE DATA SOURCING SYSTEM

This chapter describes the DSS FPGA firmwares, which are the main contribution presented by this thesis. The DSS development is part of a collaborative project that proposes an L1 trigger for the silicon tracker of the CMS experiment. The DSS were firstly idealized to emulate the incoming data (upstream) from the detector in order to test and to demonstrate the proposed trigger, case of study described by the Chapter 6. However, during its development, the system features evolved to provide a complete test system. Therefore, the DSS FPGA firmwares describe a general purpose system that can be applied to test and emulate any upstream or downstream system, as illustrated in Figure 13.

Figure 13 - (a) Complete Data Sourcing System. (b) DS Sender testing a downstream system. (c) DS Receiver testing an upstream system.



Source: Created by the author.

The DSS main purpose is to load data samples, from any format, and transfer them through high speed serial link channels, providing the received data to the receiver side. The data flow

can be synchronized and is able to transmit/receive in up to 320 Gbps per each Pulsar 2b board. A total bandwidth capability, considering an ATCA shelf operating with DSS in 12 boards, is up to 3.8 Tbps. The DSS is a debugging tool with the following functionalities:

- Change data samples without reprogramming the boards;
- Remotely reset the board;
- Control of the restart synchronous transmission;
- Check the Bit Error Rate of each channel/board;
- Check latency of each channel/board.

The DSS is composed by two similar designs, the DS Sender and the DS Receiver, which have roughly the same elements, as illustrated in Figure 13(a). However, when working separately, the DS Sender can test a downstream system and the DS Receiver can test an upstream system, as described in Figure 13(b) and (c) respectively. The DSS elements are summarized as follows:

TTC system Element able to either emulate or use an external input of signals to be propagated in the ATCA backplane. The signals used are the 40 MHz Sync clock, the BC0 periodic flag (11 MHz) used to reset MGTs and FIFOs, and the Event Counter Zero (EC0) that in this case is controlled by the user to be a starter signal. The 40 MHz clock passes through a Phase-Locked Loop (PLL) that multiplies it by 6. The resultant clock of 240 MHz is used to synchronize the internal logic of the DSS design. The combination between those signals are used to synchronize the boards in the same ATCA shelf.

IPbus User Interface It provides a direct connection between the computer and the FPGA memory blocks. The TCP/IP network is provided by an MGT connected through the ATCA backplane to an ATCA Switch. It allows to read and write data to the RAM blocks at the FPGA board at 31.25 MHz rate. The DSS is compatible with the IPbus protocol (LARREA et al., 2015). It is also possible to use write and read (W/R) operations to set and get parameters from the board, like reset signals, change tuning, measure BER and latency, among others.

DSB The Data Sourcing Board is composed by the Remote Control system, tuning parameters, 40 Groups of Data Sourcing Module (DSM), FIFOs, MGT Serial Link, BER and latency measurement systems.

Remote Control System Translates the remote control W/R from IPbus.

DSM Subset of memory reserved for each MGT channel. Each module has a Dual Port Block RAM of 4k x 32 bits memory. One port is used as User Interface clock domain while the other deals with the sync clock. The Sender side needs to receive a EC0 start signal from TTC System to transfer the data from the BRAM to the FIFO.

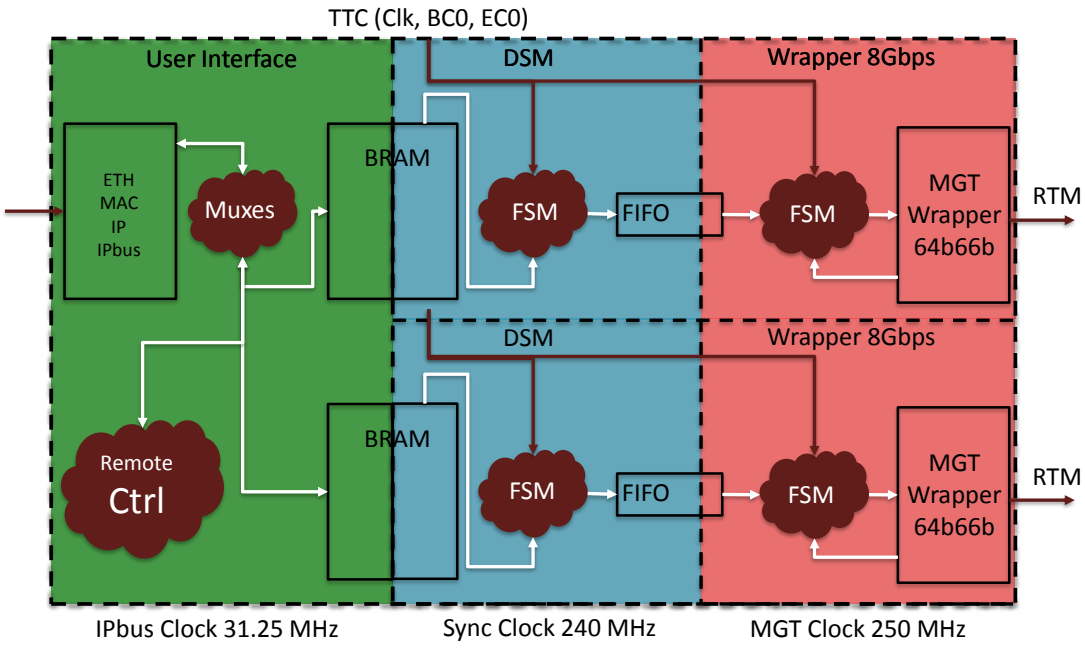
FIFO Manager The FIFO manager is located between the DSM and the MGT Channels. It also interfaces the clock domains of TTC Clock and MGT Clock. The FIFO size should be equal to the DSM size.

MGT Wrapper It is responsible for flowing data in a serial link protocol through each one of the RTM links. There are a few protocols capable of performing this task, in this work the Xilinx GT Wizard Aurora 64b/66b was used (XILINX, 2015c)(XILINX, 2015b). The data rate for this work is 8 Gbps per MGT and the clock that drives the internal parallel logic is 250 MHz. With the usage of the scrambler and descrambler, the balancing of the signals (level transition) in the channel is ensured and helps the RX clock recovery.

The difference between the DS Sender and Receiver is the data flow: the user loads the data to be transmitted in the Sender BRAMs through IPbus and, using the sync signals, can (re)load data from BRAMs to FIFOs and send it to MGTs. The DS Receiver receives the data from the channel, (re)loads the FIFO and records them in the BRAMs. After this process, the user can read the data received, measure BER and latency, and reset the system to run the scenario again.

Figure 14 shows the principal clock domains of the DS Sender FPGA design.

Figure 14 - DS Sender Clock Domains

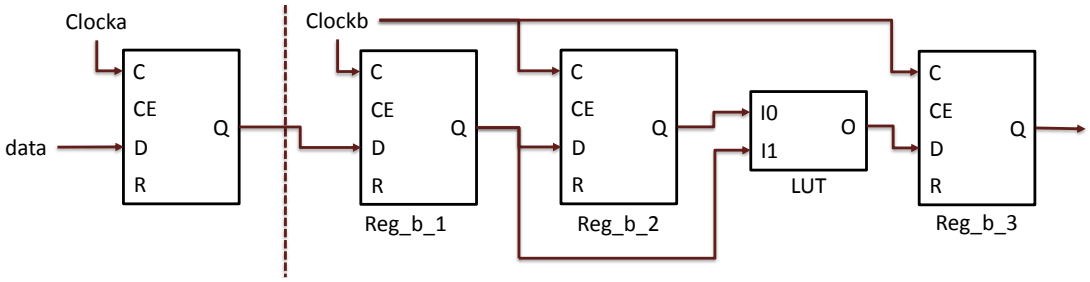


Source: Created by the author.

The DS Receiver FPGA design presents the same clock domains. Both FPGA designs used three ways to perform the crossing domain of the data and control signal.

- The DSM BRAMs perform the data interface between IPbus (31.25 MHz) and TTC clock (240 MHz).
- The FIFO Manager interfaces the data from TTC clock (240 MHz) and MGT (250 MHz) references.
- Pipelined registers are used to cross control signals used between different clock domains, as illustrated in Figure 15. In this case, false path constraints were inserted and the timing problems in that path were not considered in the timing analysis.

Figure 15 - Pipeline register to cross clock domain signals



Source: Created by the author.

The following Sections detail the data sourcing elements from sender and receiver. The VHDL files from both firmwares are listed in Appendix A.

4.1 TTC Signal Processing

The DSS system needs to share a set of signals to synchronize the processes in all boards. Those signals can either be emulated or replicated from an external source. Considering that this work is part of a HEP collaboration, by convenience we implemented a synchronization system that emulates signals coming from the LHC TTC system.

The LHC TTC system is used by the LHC experiments, like the CMS, to synchronize the electronics of all triggers of detectors present in the LHC orbit. This system delivers some signals for synchronization purpose. They are a 40 MHz Clock (bunch crossing frequency) and two control signals used by DSB. The EC0 controls the start of transmission flag in both Sender and Receiver side. The BC0 rises in every 3564 LHC clock cycles (89.1 μs) or 11 KHz, representing one LHC orbit period.

In DSS FPGA firmwares, the TTC system entity has two main functions. The first function is to get or emulate TTC signals and propagate them through the backplane in the shelf. It is possible to receive this signal from a special TTC mezzanine, the TTC FMC.

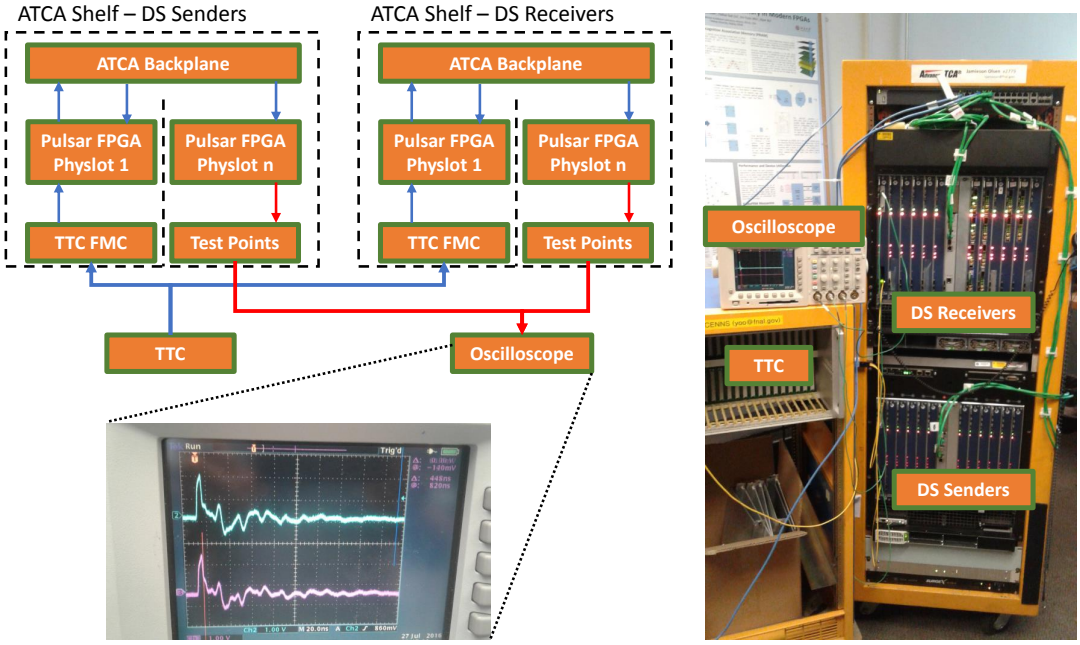
Considering a scenario where the TTC signals are not being received, the FPGA firmware can create the signals artificially using the FPGA system clock (200 MHz). This feature makes that only the Pulsars 2b boards in the same ATCA shelf be in sync, instead of a multiple shelves scenario.

Any board has the hardware capability to be the sync master, but just one of them will be choosen by the ATCA manager to do the task. This control is performed by the IPMC, that by default disables the TX drivers at the board. To enable the signal propagation, the ATCA manager must access remotely the IPMC by Telnet and perform the "bpclk" command.

The second functionality is to receive those signals from the backplane and process them. The 40 MHz clock is multipled by PLL and provides 240 MHz to the DSB internal logic. The BC0 is used by the FIFO manager. This clock domain is used by the DS Modules and FIFO manager elements.

Using the test points at Pulsar 2b boards at Physlot 1 in both shelves, as illustrated in Figure 8, it was possible to externalize the BC0 signal to an oscilloscope and to check if the signal rising happens at same time. Figure 16 shows the BC0 sampling scheme and the measurement performed, that shows the ATCA shelves synchronism.

Figure 16 - Left: BC0 sampling scheme. Right: Photo of the front view of the ATCA shelves being monitored by the oscilloscope.



Source: Created by the author. Blue arrows: TTC encoded signals. Red arrows: Decoded BC0 signal.

The same scheme shown in Figure 16 was used to sample the TTC clock and EC0 signals and check the synchronism. Some different pairs of physlots were tested and small diferences were observed [1].

4.2 IPbus User Interface

The DSS is compatible with the IPbus protocol (MANS, 2009) (FRAZIER et al., 2012) (LARREA et al., 2015), which provides a direct connection between the user personal computer (PC) and the FPGA memory blocks. The TCP/IP network is provided by a specific MGT connected through the ATCA backplane to an ATCA Switch. The IPbus protocol allows to write or read (W/R) data to the RAM blocks at the FPGA board. Both DS Sender and Receiver FPGA designs have the same IPbus component.

The IPbus protocol is a simple packet protocol that can read and write data in FPGA block memories. The protocol communication occurs between a VHDL module and software scripts in the computer. It is also possible to use W/R operations to set and get parameters from the board, like reset signals, change tuning, measure BER and latency, among others. This protocol permits data generated offline to be bufferred in the BRAM, and to send comands to control the system. Both DSB Sender and Receiver FPGA designs are compatible with basic W/R operations performed by IPbus protocol.

To make the IPbus protocol available, the Virtex-7 FPGA had to implement a Gigabit Ethernet IP core (XILINX, 2015a) to be connected at the same network as the external workstation. This ethernet connection should be attached to the MGT linked to fabric interfaces in the ATCA backplane. If the IP core is set as 1 Gbps speed, then it is free of license from Xilinx.

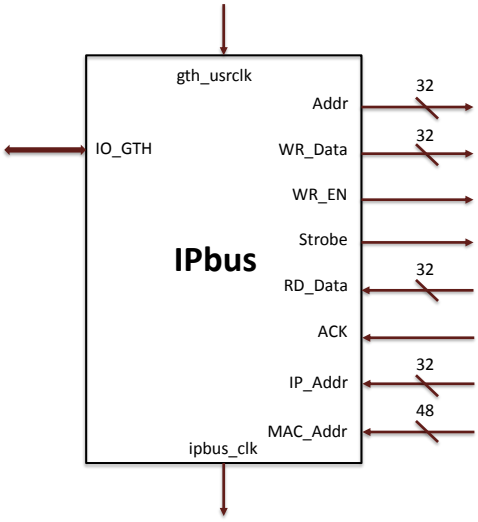
The IPbus I/O ports are illustrated in Figure 17. To perform the connections with DSB I/O ports, IPbus needs a translation for its signals. The strobe signal means W/R operation enable. When assigned together with write enable, it means a write operation. Otherwise, it means read operation.

The reference clock for the W/R operations is provided by IPbus in the frequency 31.25 MHz, which reduces the protocol bandwidth. The IPbus protocol normal mode performs only one write or read operation per TCP/IP packet. However, it is possible to use the burst mode with multiple operations per packet, that increases the bandwidth. Considering that the IPbus communication is not part of the latency measurements, the DSS does not use the burst mode transmission.

All the comands and BRAMs are available for the users that need to use a user interface

[1] The differences were related to the signal propagation through the backplane path length difference between the ATCA physical slots.

Figure 17 - IPbus I/O Ports.

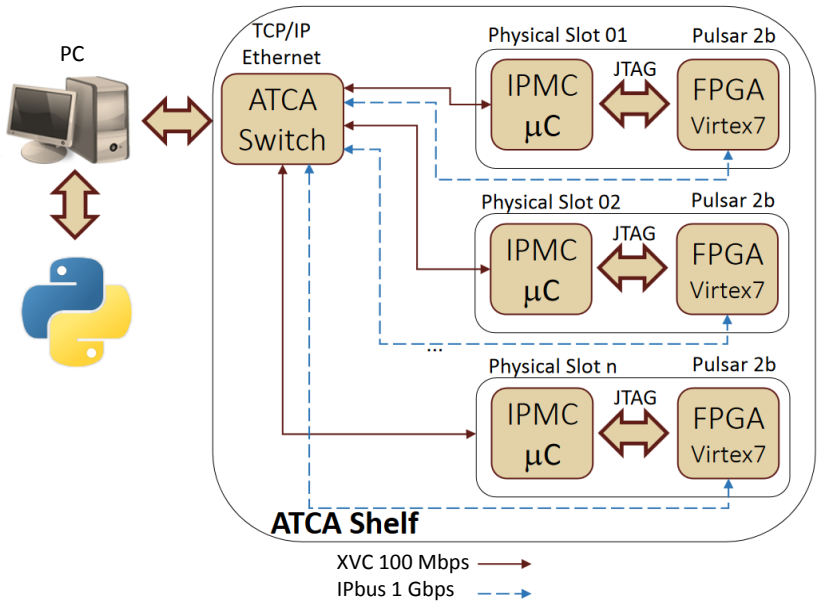


Source: Created by the author.

software at the PC. There are many languages that can be used. For this work, python scripts are used to access control commands on the DSM BRAMs.

The IPbus user interface software needs to be implemented in a computer that is connected with the ATCA shelf fabric interface (see subsection 3.1.1). The ATCA network, implemented at an ATCA switch, is interconnected to Pulsar 2b IPMC (base interface), Pulsar FPGA (fabric interface), and the user PC (both network). The ATCA network topology used is shown in Figure 18.

Figure 18 - User Interface Network Topology.



Source: Created by the author.

Through the ATCA switch, a computer can have remote access to the FPGAs first to pro-

gram/debug them using XVC (PAIVA, 2016), and after to control them through the IPbus.

The IPbus software environment, also known as Micro Hardware Access Library (μHal) (FRAZIER et al., 2012), has a transparent interface for the user that relates the memory available in a connection XML file. An example of this file applied to DS Sender FPGA design is presented below:

```
$ cat sender_connection.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<node>
    <node id="CMD" address="0x0000" permission="rw" tags="test">
        <node id="IDDLE" address="0x000" mode="single" />
        <node id="NEWIP" address="0x001" mode="single" />
        <node id="RESET" address="0x002" mode="single" />
        <node id="REARM" address="0x003" mode="single" />
        <node id="TUNING" address="0x004" mode="single" />
        <node id="COUNTER" address="0x00F" mode="single" />
    </node>
    <node id="DSM0" address="0x1000" permission="rw" mode="block" size="4096" />
    (...)
    <node id="DSM39" address="0x28000" permission="rw" mode="block" size="4096" />
</node>
$
```

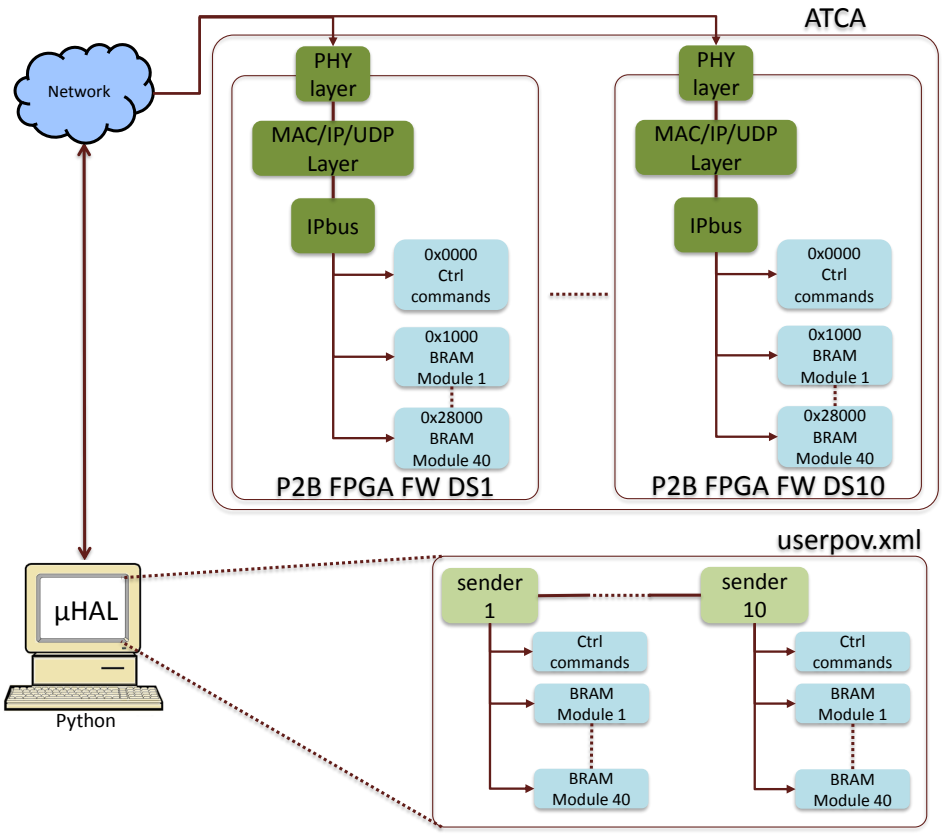
After setting a standard DS Sender connection in the XML file, the python scripts can use it to perform W/R operations, as the example below:

```
$ cat reset_sender.py
import socket
import struct
import time
import uhal # IPbus uHAL library

print "Sender Network is <ipaddress>"
print "Type the Sender IP Last Byte: "
dsb_ntw = "<ip_network>" # first 3 bytes i.e 192.168.0.
dsb_host = raw_input() # last byte of the IP address

dsb_ip = dsb_ntw+dsb_host

#Open IPbus connection
sender0 = uhal.getDevice("sender.udp.0", "chtcp-2.0://localhost:10203?target="+dsb_ip+":50001"
    , "file://sender_connection.xml")

x=sender0.getNode("CMD.RESET").read() #reset command at DS Sender Board
hw.dispatch() # send operation to the target
time.sleep(1) # sleep during reset

print "Done!"
$
```

The result of the scripts accessing the boards is presented in Section 5.1.

It provides more scalability and allows connection to different FPGAs, boards and shelves. The FNAL ATCA setup with DSS in the IPbus user point of view is illustrated in Figure 19.

Figure 19 - IPbus user point of view.



Source: Created by the author.

4.3 Data Sourcing Board

The DSB component is different between DS Sender and Receiver designs. Those components were created to allow that DSS be able to test itself when they are connected between each other. However, when acting separately, DS Sender can emulate an upstream system and DS Receiver can receive results from upstream. Those entities interconnect the subcomponents User Interface, BRAMs, state machines, link checkers, tune parameters, FIFOs and GTH wrappers. The IPbus interface connections with DSB are:

RX_Data 32 bits for writing operations at DSB BRAMs;

TX_Data 32 bits for reading operations at DSB BRAMs;

RX_Addr 18 bits for memory addressing;

RX_WREN 1 bit flag for write enable;

TX_RDEN 1 bit flag for read enable;

Ready 1 bit flag that acknowledges the reception/process of W/R operations performed by IPbus;

IPbusclk 31.25 MHz.

The most significant 6 bits from RX_Addr are used by muxes/demuxes that makes the DSB appear as a big block memory. This transparency hides the DSB internal components, that are a remote control system and 40 DSM and the indexing is performed as follows:

RX_Adress = 0b000000 The muxes redirects the ipbus interface to Remote Control element.

RX_Adress = 0b000001 The muxes redirects the ipbus interface to DSB # 1.

RX_Adress = 0b<dsb_idx> The muxes redirects the ipbus interface to DSB # <dsb_idx>.

RX_Adress = 0b101000 The muxes redirects the ipbus interface to DSB # 40.

Each one of the subcomponents of DSB are different between sender and receiver.

4.3.1 Remote Control System

The most significant 6 bits equal zero redirects the connection to remote control system. This component emulates a block RAM behaviour for IPbus component, although it translates the W/R operations in command/debug signals. Among the commands performed by the DS Sender and Receiver it is possible to find:

New IP Overwrites the ethernet interface block default IP address. The updated address is the IPbus new address.

Reset Inserts a reset to all state machines and signals inside DSB.

Tune Parameters It is possible to change the MGT channel tuning parameters without reprogramming the board.

BER Mode Used to start the BER measurement system of all channels. When the command is activated, the DS Sender changes the channel data input to a known word. At receiver side, it starts the bit error measurement. When BER Mode is deactivated (default), DSB performs the normal data flow.

Counter Used only at the DS Sender, it tells how many words the DSM needs to send to the MGT in the normal data flow.

BER Used only at the DS Receiver, it allows to read BER measurement from each channel.

Latency Used only at the DS Receiver, it allows to read Latency measurement from each channel.

The range of possible command addresses goes up to 4096. For now, the DS Sender has 7 commands available as follows:

- 0x001 - New ip;
- 0x002 - Reset;
- 0x003 - Rearm;
- 0x004 - TX tune settings;
- 0x005 - DSB Mode/ BER Mode;
- 0x006 - Start/Stop Streaming DSB Mode;
- 0x00f - 32 bits word Counter;
- 0x010 to 0xfff - not supported.

Moreover, the DS Receiver has 85 commands available as follows:

- 0x001 - new ip;
- 0x002 - reset;
- 0x003 - rearm;
- 0x004 - RX tune settings;
- 0x005 - DSB Mode/ BER Mode;
- 0x006 to 0x100 - not supported;
- 0x101 to 0x128 - BER measurement reading from channels # 1-40;
- 0x129 to 0x200 - not supported;
- 0x201 to 0x228 - Latency measurement in MGT clock cycles from channels # 1-40;
- 0x229 to 0xfff - not supported.

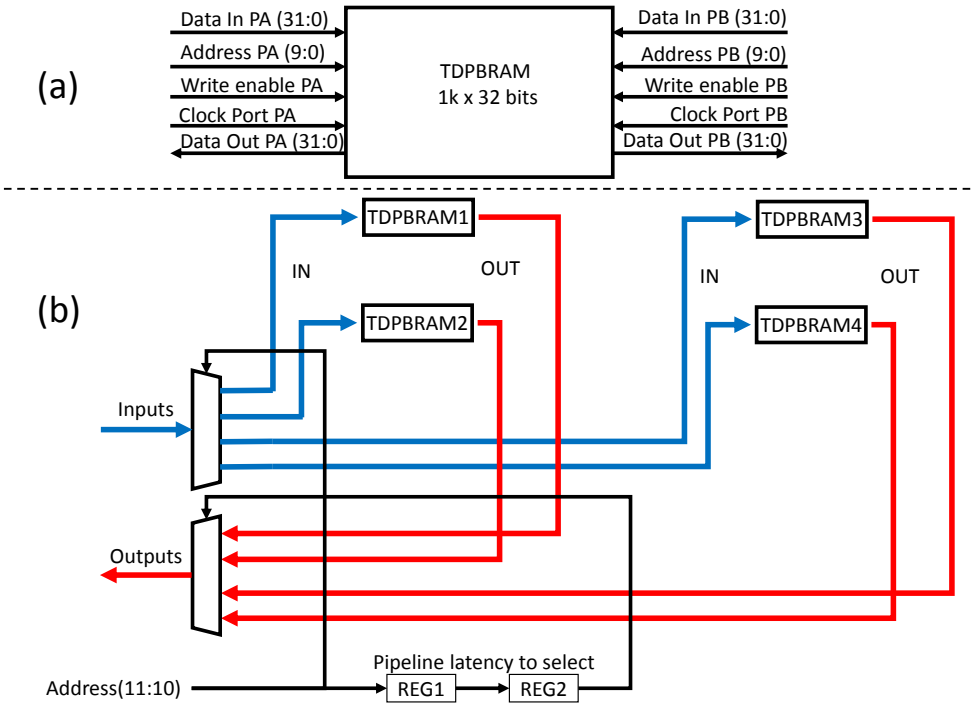
There are addresses available to insert more commands in the future.

4.3.2 Data Sourcing Modules

The most significant (MS) 6 bits index redirects for one of 40 DSM available. Each DSM has 4096 addresses (12 bits) available in the BRAMs. Each address holds 32 bits of data. Each DSM has the capability to hold 131072 bits distributed in 4 block memories.

The modules have 4 True Dual Port Block RAMs (TDPBRAM) (XILINX, 2013) as illustrated in Figure 20.

Figure 20 - DSM TDPBRAM Ports. (a) TDPBRAM I/O Definitions. (b) Mux/Demux selection is related to the provided address in one of the ports.



Source: Created by the author.

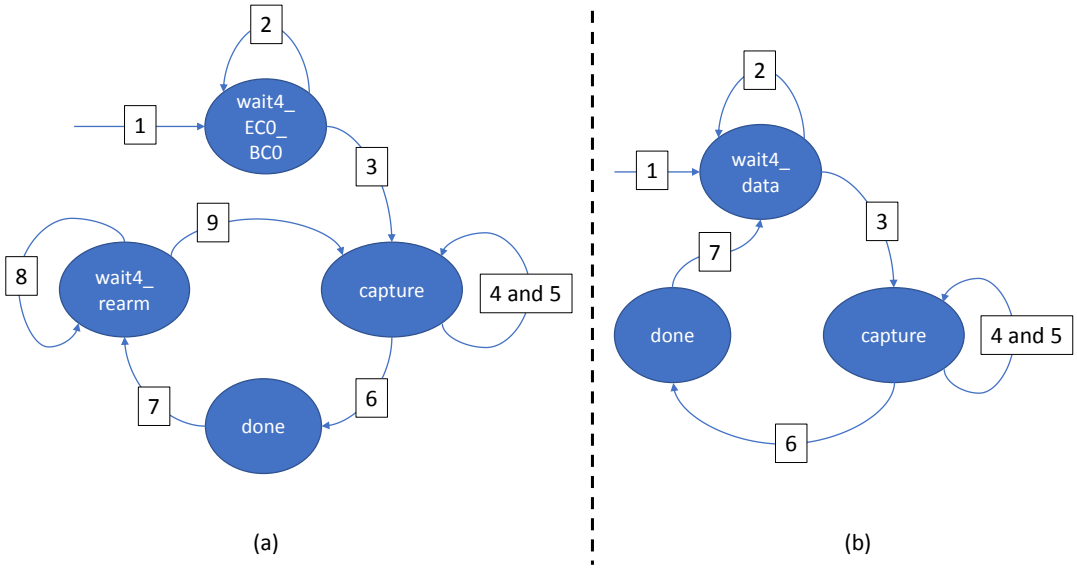
The dual ports are available as Port A (PA) and Port B (PB) and the I/Os are defined in Figure 20(a). Figure 20(b) shows that the 2 most significant bits are used as mux/demux in the inputs and outputs from both ports. The inputs mux selection is performed without latency. However, the data output, related to the BRAM reading instrisic delay, is pipelined to insert 2 clock cycles of latency in the demux selection. The block memory intrinsic delay is a latency between address input and data output (XILINX, 2013).

In the FPGA design, the PA and PB interfaces are connected to IPbus and DSM finite state machine (FSM), respectively. All the logic between DSM and the FIFOs are guided by the TTC clock (240 MHz) that ensures that the behaviour is synchronized between boards in different shelves.

In the DS Sender and Receiver designs, the DSM FSM's are different as illustrated in Figure

21. The state transitions indexes presented in Figure 21 are correlated with the indexes described in the Tables 2 and 3.

Figure 21 - DSM FSM's. (a) DS Sender FSM. (b) DS Receiver FSM.



Source: Created by the author.

Table 2 - DS Sender FSM behaviour.

Current State	Conditions	Action	Next State	IDX
Any	reset = 0b1	Initializes all signals. Sets DSM 32 bits word output to idle 0x00000000.	wait4_EC0 _BC0	1
wait4_EC0 _BC0	EC0 = 0b0	Initializes all signals. Sets DSM 32 bits word output to idle 0x00000000.	wait4_EC0 _BC0	2
	EC0 flag = 0b1 and first BC0 = 0b1 after it	Increments BRAM port B address. Sets words sent counter to zero. DSM 32 bits word output stays idle.	capture	3
capture	words_sent < user_counter and first clock cycle	Increments BRAM port B address. DSM 32 bits word output stays idle.	capture	4
	words_sent < user_counter	Increments BRAM port B address. DSM 32 bits word output receives BRAM PB output. Increments words sent counter. Sets FIFO write enable = 0b1.	capture	5
	words_sent = user_counter	Stops BRAM port B address increment. DSM 32 bits word output receives BRAM PB output. Sets FIFO write enable = 0b1.	done	6
Done	Any	Sets DSM 32 bits word output to idle 0x00000000. Sets BRAM port B address to zero. Sets FIFO write enable = 0b0.	wait4_ rearm	7
wait4_ rearm	rearm = 0b0	Does nothing.	wait4_ rearm	8
	rearm = 0b1	Sets words sent counter to zero. Increments BRAM port B address. DSM 32 bits word output stays idle.	capture	9

Source: Created by the Author.

Table 3 - DS Receiver FSM behaviour.

Current State	Condition	Action	Next State	IDX
Any	reset = 0b1	Initializes all signals.	wait4_data	1
wait4_data	write enable from FIFO = 0b0 or stop_record = 0b1	Sets BRAM PB write enable = 0b1.	wait4_data	2
	write enable from FIFO = 0b1 and stop_record = 0b0	Sets BRAM PB write enable = 0b1. Sets BRAM data input receives FIFO output. Increments words written counter.	capture	3
capture	write enable from FIFO = 0b1 and words_written < maximum	Sets BRAM PB write enable = 0b1. Sets BRAM data input receives FIFO output. Increments BRAM PB address. Increments words written counter.	capture	4
	write enable from FIFO = 0b1 and words_written = maximum	Sets BRAM PB write enable = 0b0. Sets stop record to 0b1.	Done	5
	write enable from FIFO = 0b0	Sets BRAM PB write enable = 0b0. Increments BRAM PB address.	Done	6
Done	Any	BRAM PB write enable = 0b0.	wait4_data	7

Source: Created by the Author.

While the sender reads BRAM PB and writes data in to the FIFO, the receiver reads from FIFO and writes at BRAM PB. The reading process should consider the BRAM reading latency. To do so, they use the same pipeline register scheme decribed in Figure 20(b).

4.3.3 FIFO Manager

The FIFOs in the data flow are placed to cross the data between the clock domains TTC 240 MHz and the MGT clock 250 MHz. In both sender and receiver, the designs have to manage the write and read operations to do not over flow or drain out the data at the FIFO. To do so, the design used the approach to do not allow read and write operations to happen at the same time.

At sender and receiver the FIFO Manager has different behaviour. The DSM at the sender writes at the FIFO using the 240 MHz clock. The FIFO manager should read the data to send it through the transceiver using the MGT reference clock (250 MHz). The opposite data flow occurs at the receiver.

At the sender side, the FIFO manager starts its execution after the detection of the falling edge of FIFO write enable that comes from DSM. This step means that the writting process performed by DSM has finished. At the next BC0 rising edge the FIFO starts the transmission.

The data words have 32 bits of width. To identify and to allign the reception, a starter frame 0xffffffff is inserted at the beggining of the data to be sent. The starter frame divides groups of 16 data words plus optional number of null words. The null words can be used to adequate the

throughput of each link. However, the DS Receiver should be set in the same way, maintaining the transmission consistency.

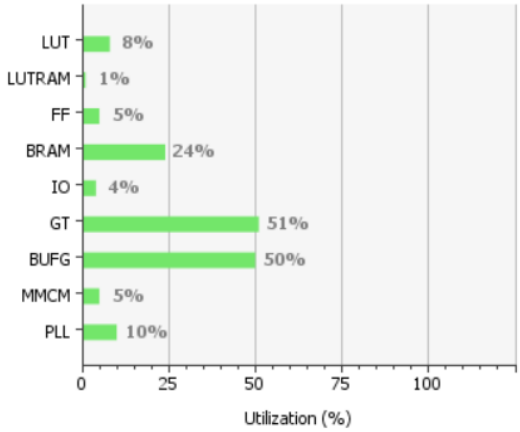
The FIFO manager at the receiver waits the starter frame starts its FIFO writing process. When it appears from MGT data output, the manager should rise a firstword flag for latency measurement system. Moreover, it starts counting 16 words to write in the FIFO. After the writing of the data words group in the FIFO, the manager rises a signal that allows the FIFO reading and the DSM BRAM writing. Moreover, it discards a predefined number of null words before another starter frame arrival. The BRAM writing process procceds until the BRAM maximum capacity is achieved.

4.3.4 Normal Data Flow Summary

In a normal data flow, the data is loaded into the DS Sender shelf using IPbus, as described in Subsection 4.2. A file can be used to load data samples to be transmitted by each MGTs connected to the RTMs in the shelf. After loading the data, the user starts the data flow process with the sync EC0 signal, as described in Subsection 4.1. The normal data is transfered and can be compared at the receiver shelf to check errors. However, for an evaluation of the integrity of the links the amount of data available from the BRAM are not statiscally representative. Thus, the DS Sender and Receiver designs provide the BER data flow as alternative for the integrity evaluation (see Subsection 4.3.6).

The same amount of memory blocks used for DSMs are necessary for the FIFOs. So, from the FPGA utilization, as shown in Figure 22, there are 8 BRAMs which are used per MGT channel connected with RTM what results in a total of 320 BRAMs used in the sender or receiver designs. Considering the Virtex 7 FPGA block RAM resources available, there is room to scale the memory capacity at least 4 times.

Figure 22 - Virtex 7 FPGA utilization by DS Sender design.



Resource	Utilization	Available	Utilization %
LUT	33391	433200	7.71
LUTRAM	51	174200	0.03
FF	39488	866400	4.56
BRAM	348.50	1470	23.71
IO	24	600	4.00
GT	41	80	51.25
BUFG	16	32	50.00
MMCM	1	20	5.00
PLL	2	20	10.00

Source: Created by the author.

The DS normal data flow from the sender and receiver is summarized in Figures 23 and 24. The figures were taken by simulation at Vivado software.

The wrapper interface at the sender and receiver considers the Gearbox pauses of the GT Wizard for Aurora 64b66b Protocol (see Section 4.4).

The firstword signal is used by the latency measurement system. Each MGT has the latency measured separately. More details in the next Subsection.

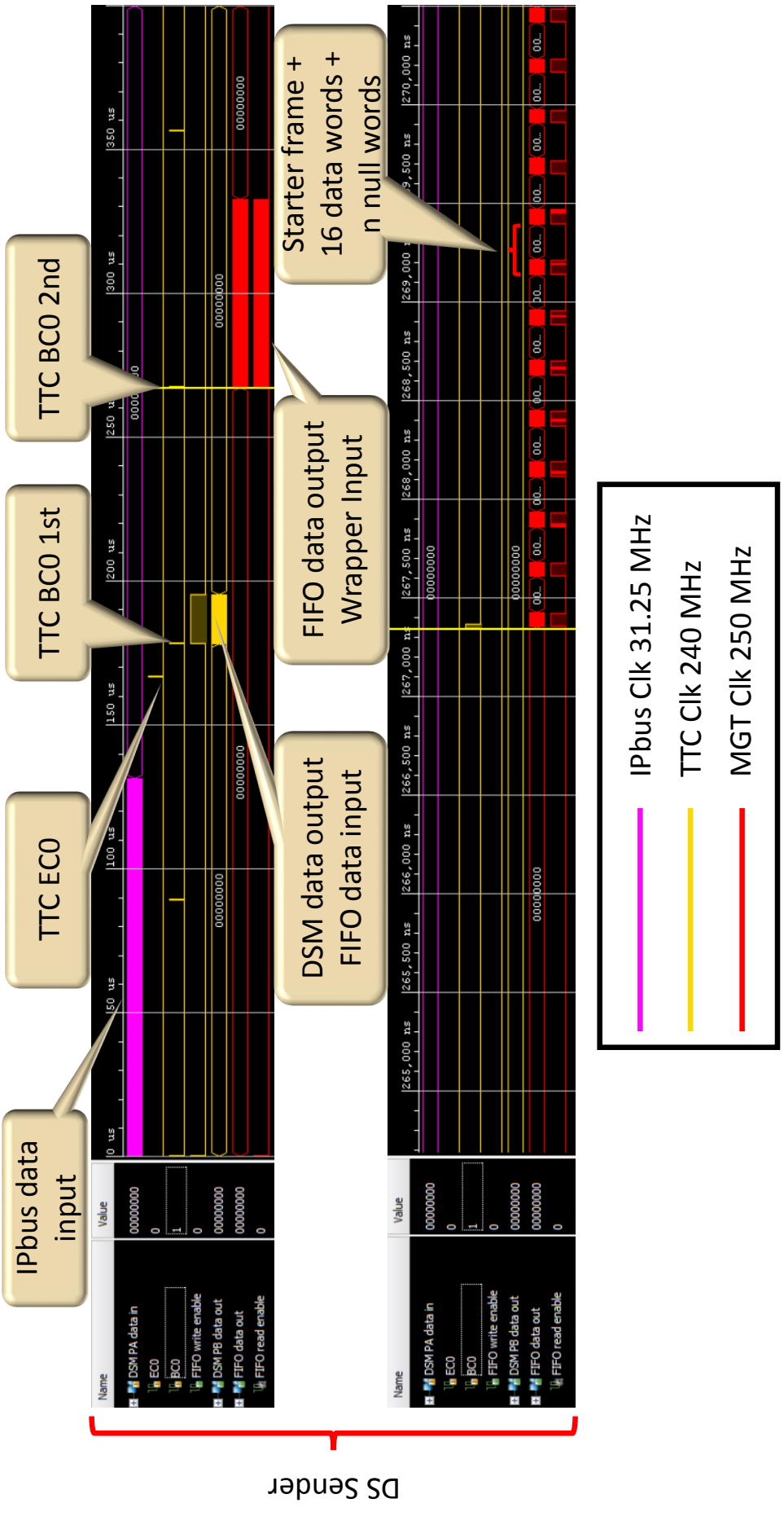


Figure 23 - Sender data flow.

Source: Created by the author.

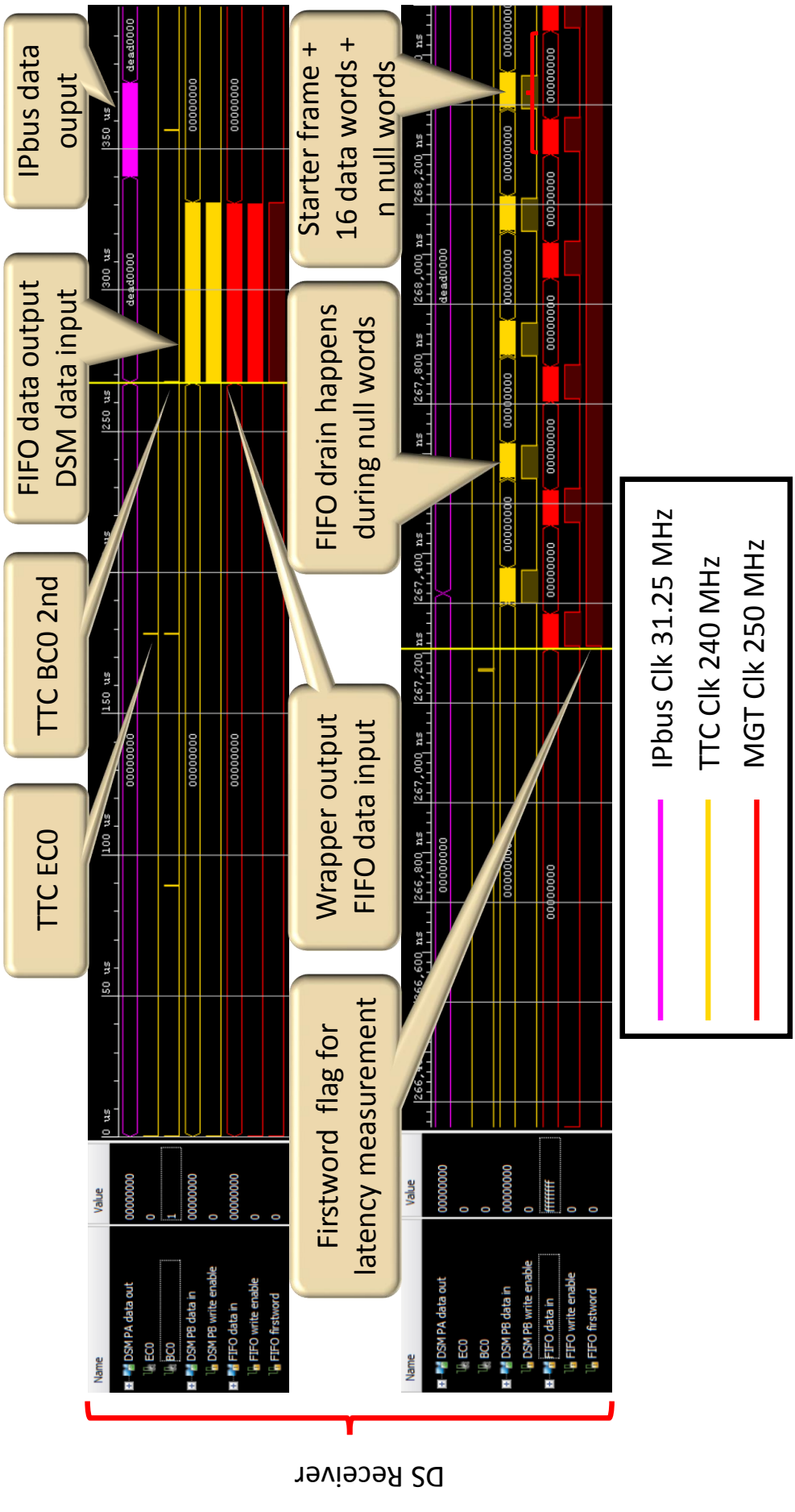


Figure 24 - Receiver data flow.

Source: Created by the author.

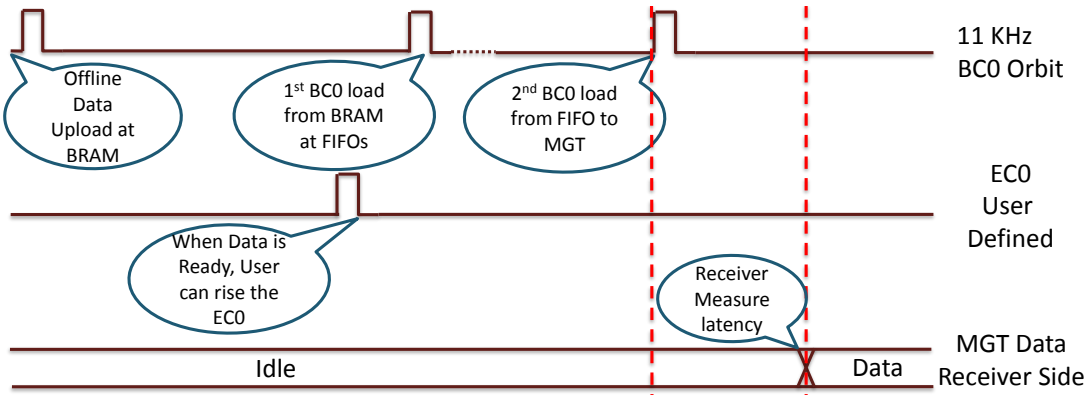
4.3.5 Latency Measurement

The hardware setup, as described in Chapter 3, uses two ATCA shelves with 12 Pulsar 2b boards in each one, as shown in Figure 3. The bottom ATCA shelf is programmed with DS Sender, and the upper one with the receiver, both operating with links at 8 Gbps speed rate. The latency is directly related with the speed rate.

All TTC signals are received by one board in each shelf. That board is responsible of receiving the TTC signals and replicate them through the backplane, ensuring that all boards from both shelves are synchronized.

The EC0 signal is received by the Pulsar 2b boards and triggers the transfer of the data loaded in each of the DSMs, as described in subsection 4.3. The receivers receive the same sync signals as the senders and use them for the latency measurements, as illustrated in Figure 25.

Figure 25 - TTC Synchronization Signals usage.



Source: Created by the author.

The latency is measured in 250 MHz clock cycles and can be read by the IPbus user interface. The clock domain crossing between MGT and IPbus reference is performed by FIFO usage. With a reset, the user can repeat the process and measure the latency again.

4.3.6 Link BER Checker

In the normal data flow of the current design, the amount of data is 131 kbits per channel. The BER mode data flow sends a greater amount for a better evaluation.

To have a stable link with high speed serial link protocols, sometimes it is required to tune transmitter and receiver so attenuation and interference can be compensated. It is possible to use the Xilinx IBERT (XILINX, 2016a) to evaluate the link reliability. So, the FNAL RTM links were evaluated using the Auto Tuning System, as described in Subsection 4.4.1.

The DSS uses an integrity evaluation tool to allow the link BER check without reprogramming the boards. However, the evaluation has influence not only in the optical link condition but also on how the enconding of the serial link protocol can deal with that.

The BER data flow can be activated by the user in both sender and receiver. This is possible using the "BER mode" command, as described in Section 4.2. After this, a mux will replace the normal flow by the BER flow as input of the wrapper in the sender side. In this case, the DS Sender loads a fixed 32 bits word (0xffff0000) in each channel. The MGT TX encodes the word and serializes it to transfer.

After the deserialization performed by the MGT RX, the DS Receiver counts the bit errors presented in each word. In the receiver, the "BER mode" starts and the data words are received until it achieves 10 Tbits (10^{13}). The time window needed is roughly seven minutes. It is possible to increase the measurement resolution by increasing the total number of bits sent/received. However, the measurement time window increases by orders of magnitude, i.e 70 minutes to provide 10^{14} bits or even 700 minutes to 10^{15} bits.

Thus, the BER measurement value is available for user reading through IPbus. The clock domain crossing between MGT and IPbus reference is performed by FIFO usage. With a reset, the user can repeat the BER mode process.

4.4 Wrapper Protocol

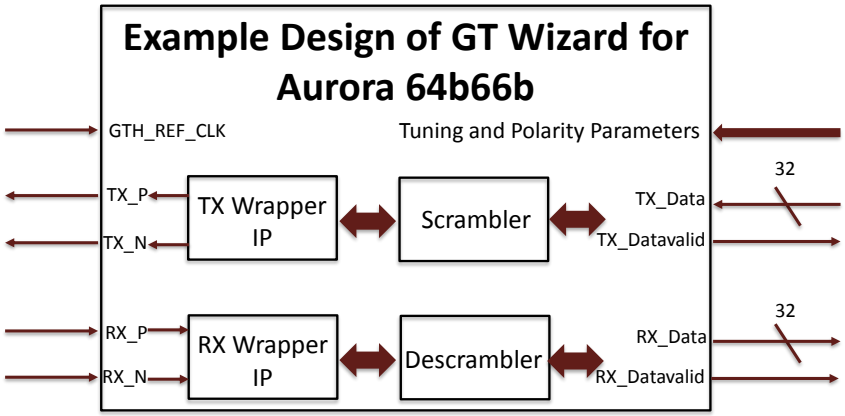
The wrapper is responsible for flowing the data in a serial link protocol through each one of the RTM links. There are some protocols that can execute this task. In this work, the Xilinx GT Wizard for Aurora 64b/66b IP core together with its example design were used. The data rate for this work is 8 Gbps per MGT, and the clock that drives the internal parallel logic is 250 MHz.

The IP example design used was changed only to externalize the signals that are employed for DSB purpose, such as 32 bits Data I/O, Gearbox data valid, tuning parameters, MGTs polarity and TX or RX user clock. The example design I/O ports (XILINX, 2015c) are illustrated in Figure 26. The tuning and polarity setting are described in Subsection 4.4.2.

The 64b/66b encoding has a 3.125% overhead in the line rate. It has a Gearbox (XILINX, 2015c) that uses a signal flag called data valid that informs when input/output data can be transmitted to the wrapper. This pause is periodic and non-deterministic between the MGTs. It stays high for 32 clock cycles and low for 1 clock cycle. In the low state, it does not insert or sample the data.

The MGT user clock is isolated near to the transceivers using the FIFO manager. The TTC 240 MHz clock takes care of the DS internal logic to ensure a synchronization at this part. The

Figure 26 - GT Wizard for 64b66b Aurora IP Example Design IO used.



Source: Created by the author.

latency variation is related to the Gearbox pauses, cross domain signals between TTC clock and MGT clock, or even optical signal propagation in different length cables.

Everytime when the sender is idle in the normal data flow, it sends zeros as data input to the wrapper link. However, to balance the transitions between zeros and ones symbols, any word to be sent passes through a scrambling before it goes to the serialization at MGT. In the receiver side, the balanced word is received and descrambled, recovering the idle words to the receiver. A similar process happens to send and receive for non idle words.

With the usage of the scrambler and descrambler the transition of the signals in the channel is ensured. This improves the RX clock recovery system.

4.4.1 Auto Tuning System

The tune parameters are used to compensate signal distortions and to reduce the bit error rate. The signal waveform compensation can be applied at emphasis gain or at transceiver differential swing voltage calibration (XILINX, 2012).

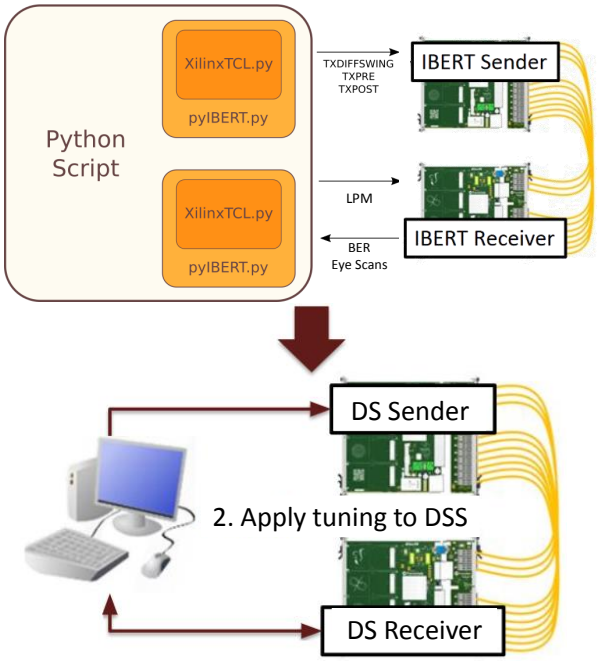
The transceiver emphasis is applied to help the bit transition identification. When the communication keeps the same bit without transition, either 0-0 or 1-1, the swing voltage tends to decrease and the receiver can perform a sampling error. The emphasis will mitigate the reduction at the beginning of the waveform period, right at the moment where the receiver should sample. The same tune effect can happen trying to increase the transceiver differential swing voltage (XILINX, 2012).

Taking into account that each channel in an application has different paths for the signals, such as board trace, backplane or cable, then the wave distortion can be different between those channels. Therefore, it is important to have tune optimization dedicated for each channel (XILINX, 2012).

Considering Xilinx FPGAs, it is possible to use the Integrated Bit Error Ratio Tester (I-BERT) to evaluate the reliability of the link. With this tool the user can apply PRBS to stress the channel and then measure the BER and/or generate eye scans. Therefore, IBERT also provides tools for setting the tune parameters automatically. However, this is only possible on applications where the target FPGAs are in the same JTAG chain or the target link is in loopback mode. The hardware setup used in this work connects FPGAs placed in different JTAG chains, which demands a manual tuning of the channels.

To optimize this process, some scripts were created to allow an automatic tuning of the links, as illustrated in the top of Figure 27. The Auto Tuning System (SPRACE, 2017) is composed by two classes:

Figure 27 - Auto Tuning Scripts Scheme and Settings at Data Sourcing System.



Source: Created by the author.

- XilinxTCL.py, which implements commands interface between python and Xilinx tcl language;
- pyIBERT.py, which uses XilinxTCL.py to implement specific commands related to Xilinx IBERT tests.

Basically, a python script uses these two classes to communicate with Vivado and to set tune values on transmitter and receiver. The performance of the link is measured by the same script, which checks the channel stability and performance. Different test configurations were used in order to find an optimal tune setup.

In order to callibrate the hardware setup used by this work, we used this Auto Tuning System together with IBERT to find a set of tuning parameters to be applied in all boards of the shelves. The parameters investigated in this work together with their initial values are listed in Table 4.

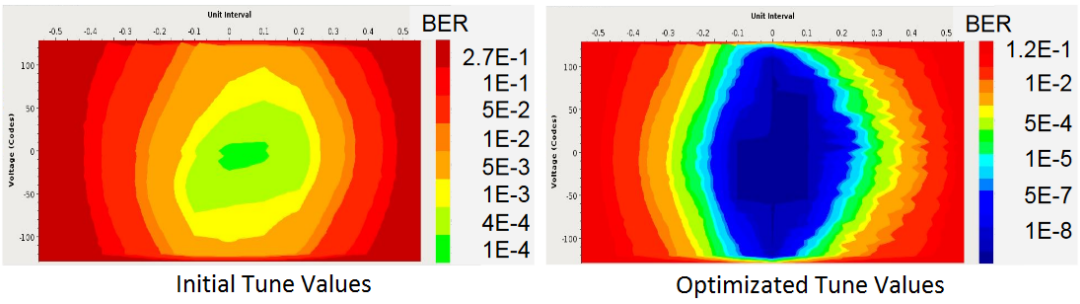
Table 4 - Tune Parameters and their initial values.

Tune Parameters	Initial Values	Optimization Values
TX Diff Swing Voltage	807 mV	807-1092 mV
TX Pre Cursor Emphasis	0 dB	0-4.44 dB
TX Post Cursor Emphasis	0 dB	0-1.94 dB
RX Equalizer	LPM	LPM

Source: Created by the author.

With this setup the channels were stressed by IBERT using the PRBS7. The tuning parameters were optimized separately for each channel, according to its BER and eye scans measured. The difference between the eye scan before and after the tuning is illustrated in Figure 28. For the TX parameters, different values were found and the range of post-optimized values are presented in Table 4. In RX side, the Low-Power Mode (LPM) equalizer showed better performance in all cases.

Figure 28 - Difference between eye scans before and after the tuning optimization process.



Source: Created by the author.

Thus, the DS shelf level tests used two sets of tuning parameters in the BER tests (see section 5.2), one with the initial tune parameters values applied to all channels, the other with parameters optimized by Auto Tuning tests. It is important to state that Auto Tuning System is not one of the contributions of this thesis. It optimized the tuning parameters to improve the Fermilab ATCA setup performance. The parameter settings that were found by Auto Tuning System were used in DS BER tests.

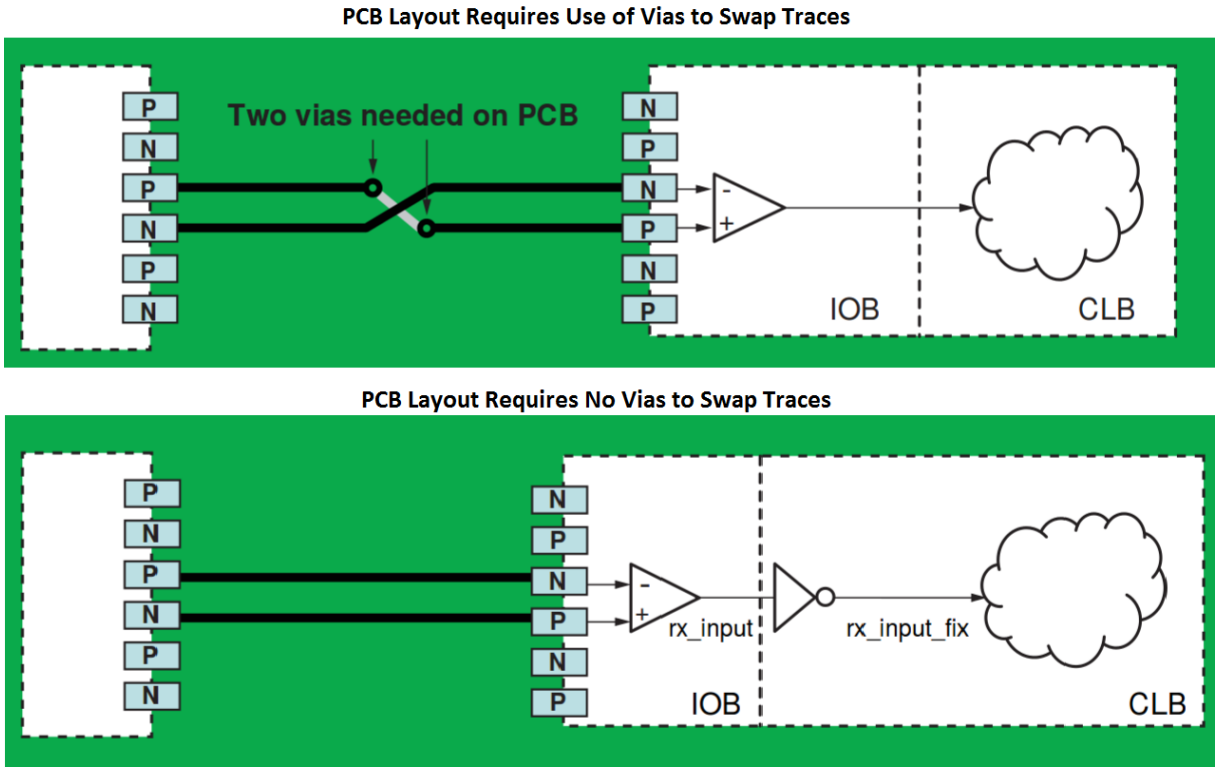
4.4.2 Tuning Parameters

This entity connects directly the tune and polarity settings of the 40 MGT wrappers.

In some situations it is necessary to invert the MGT signals (tx_p, tx_n, rx_p, rx_n) in the board layout to make the PCB design possible, as shown in Figure 29.

This technique can be usefull for PCBs that contain hundreds of I/Os to be routed and the impedance matching to be considered. So, the developer can invert the positive and negative pins in the layout process and recommend the FPGA firmware developer to revert the pins again in the MGT. In this case, the signal is be inverted in the FPGA design through the addition of a "not gate".

Figure 29 - PCB Layout with signals reversing.



Source: Created by the author.

The link polarity is set hardcoded in the FPGA design considering the Pulsar 2b board electronic schematic.

The DS links tuning parameters, separated in TX for sender side and RX for receiver side firmwares, that the FPGA design can change are:

TX Differential Swing Voltage There are 16 possible voltage setups for this parameter.

TX Pre Cursor Emphasis There are 20 decibels emphasis setups for this parameter.

TX Post Cursor Emphasis There are 32 decibels emphasis setups for this parameter.

RX DFE/LPM Equalizers There are 2 possible equalizers that can be set (in our approach).

The tune parameters effects in described in the Subsection 4.4.1.

There are two types of adaptive filtering available to the GTH receiver depending on system level trade-offs between power and performance. The LPM is optimized for power with lower channel loss (XILINX, 2015c). The Decision Feedback Equalizer (DFE) allows better compensation of transmission channel losses by providing a closer adjustment of filter parameters than when using a linear equalizer.

The DSS already implemented different hardcoded settings for all the link tune parameters. Moreover, through IPbus it is possible to remotely change the current settings without reprogramming the board. A FIFO is used as cross domain scheme between the IPbus clock and MGT user clock, as explained before in this Chapter.

4.5 Chapter Summary

This Chapter presented the details about the FPGA firmwares from DSS. Aspects of the system behaviour will be explored and demonstrated by the Chapter 5.

5 TEST AND RESULTS

This Chapter shows the tests and results related to the DS Sender and the DS Receiver interconnected. First in Section 5.1, tests and command results are shown for board level tests. Then, Section 5.2 presents the shelf level tests and results.

The system can be accessed remotely for programming and debugging from Vivado through the internet and XVC protocol directly (RAMALHO et al., 2015). To have access to the IPbus network, the user needs to log at the linux gateway and perform the python scripts from there.

The IPbus network has one Virtual Local Area Network (VLAN) set for each ATCA shelf. Thus, the sender and receiver FPGA designs have different IP networks.

- DS Receivers IPbus network: 192.168.101.0/24.
- DS Receivers default address: 192.168.101.14.
- DS Senders IPbus network: 192.168.102.0/24.
- DS Sender default address: 192.168.102.14.

All DS Sender has the same FPGA bitstream generated and the default IP addresses need to be changed to avoid IP duplication conflict. The IP address assignment criterion used was to replace the default byte 0d14 (0xe) by a byte composed of the sum 100 plus the number of the physlot of the board. For example the DS Sender board placed at physlot 1 has the IP changed to 192.168.102.101. The same strategy is used for the DS Receivers.

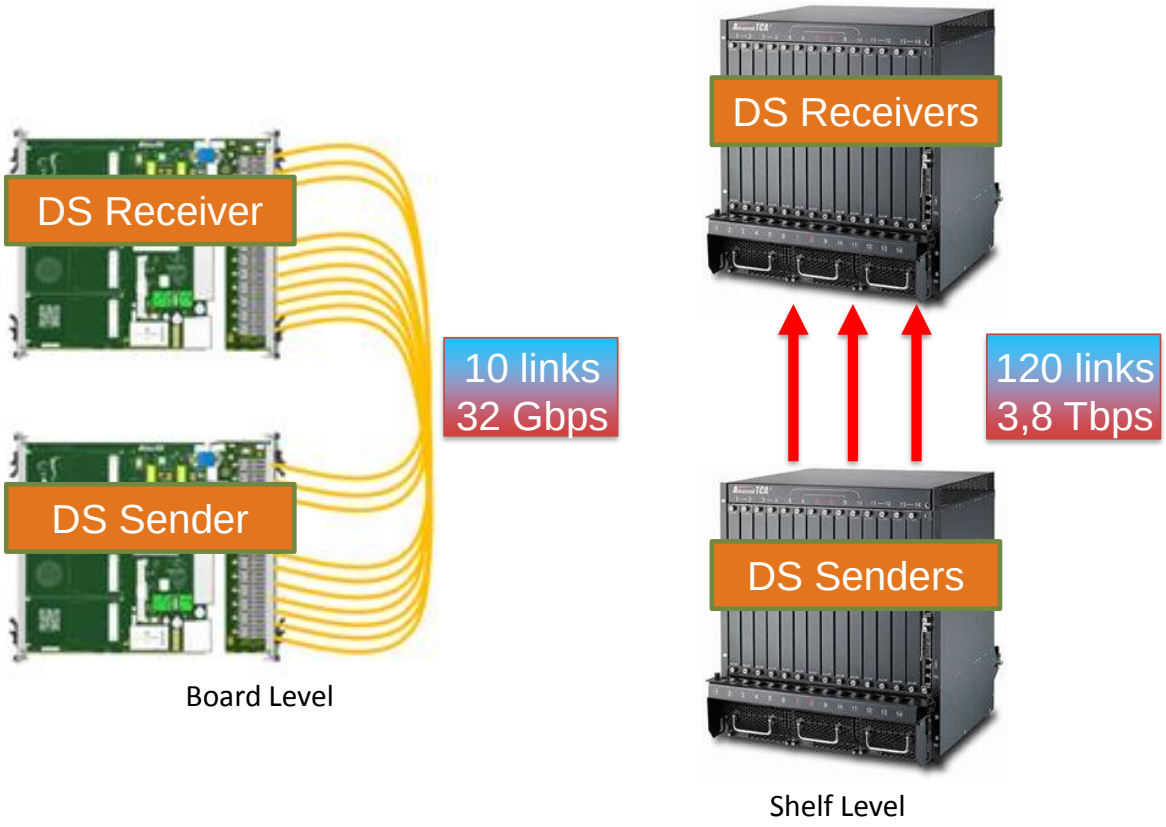
The IPbus python scripts are listed at Appendix B.

5.1 Data Sourcing Board Level Tests

This Section presents IPbus user interface operations in the software side to prepare and test the Data Sourcing setup. The result shown represents the management of the FPGA signals, like W/R in the BRAMs or control signal generation.

The DS board level setup uses two Pulsar 2b boards interconnected by 10 QSFP+ optical cables from the RTMs. One of the boards is programmed with the DS Sender design and the other with the receiver from another board, as illustrated in Figure 30. The tests scale to a shelf level setup.

Figure 30 - Data Sourcing Board and Shelf Level Testing.



Source: Created by the author.

The first step of the DSS setup is to program the FPGA design and to set IPbus IP address board by board. This process avoids IP address duplication inside the crate network. The result of the IPbus software script run is:

```
1 $ python ipbus/Change_IP_Sen.py
2 Sender IP Address from 192.168.102.14
3 New IP...
4 Last Byte =
5 101
6 101
7 Checking new IP...
8 Sender IP Address changed to 192.168.102.101 !
9 $
```

The next important step is to load the data samples in the DS Senders. The example below uses a file with test data for all 40 DSMs available in one board. The data amount per GTH channel is 512 bits splitted in 16 words. The counter to be written needs to tell to DSM how many words are intended to be sent. A greater amount of data can be used. The same process is used to load data in several boards in the DS Sender shelf.

```
1 $ python ipbus/WriteDummy_Sender.py
2 Sender Network is 192.168.102.0
3 Type the Sender IP Last Byte:
4 101
5 Loading data to Sender  192.168.102.101
```

```
6
7 Please Wait .
8 Data sample  0 in GTH X 0Y0
9
10 ...
11
12 Data sample 33 in GTH X 0Y33
13
14 Data sample 34 in GTH X 1Y0
15
16 ...
17
18 Data sample 39 in GTH X 1Y5
19
20 Please Wait ...
21 Counter set to send: 16 words
22 Done! Data loaded at Sender 192.168.102.101
23 $
```

With the end of data loading the user can trigger the scenario using the TTC, as described in Section 3.3, or continue to prepare other boards for a large data flow scenario.

The XDAQ web server is provided by the linux machine connected to TTC VME board using USB cable, as illustrated before in Figure 3. The XDAQ Interface allows the configuration of the clock, BC0 and EC0 signals used by DSS.

After sending the EC0 to the DS Senders, the user can start the reading process at the DS Receivers. The reading process is important to check if the data received is the same as sent. It is possible to count the words with errors. The same file used to load the senders is used to check the reception.

```
1 $ python ipbus/ReadData_Receiver.py
2 Receiver Network is 192.168.101.0
3 Type the Receiver IP Last Byte:
4 101
5 Read Data from Receiver  192.168.101.101
6 Insert the Board Number: 0
7 Board00.txt File Selected
8 Please Wait ...
9 Frame Checking
10 Module 0
11 ...
12 Module 39
13
14 32 bit words with errors at Board =  0
15 $
```

The user can rearm the senders to transfer the data samples, as shown below:

```
1 $ python ipbus/ResetRearm_Sender.py
2 Sender Network is 192.168.102.0
3 Type the Sender IP Last Byte:
4 101
5 Done!
6 $
```

After the normal data flow is triggered the user can read the latency, as described in Subsection 4.3.5. The latency measurement results, shown below, presents a small differences between the channels. The difference is up to 4 clock cycles at 250 MHz and it can be explained by the 64b/66b Gearbox pauses from the Serial Link protocol used (see Section 4.4).

```
$ python ipbus/Latency_allgth.py
Receiver Network is 192.168.101.0
Type the IP Last Byte:
101
Receiver Board:  192.168.101.101
Please Wait...

GTH X0Y 0  Latency: 136  ns
GTH X0Y 1  Latency: 140  ns
GTH X0Y 2  Latency: 140  ns
...
GTH X0Y 32  Latency: 152  ns
GTH X0Y 33  Latency: 152  ns
GTH X1Y 0  Latency: 148  ns
GTH X1Y 1  Latency: 148  ns
...
GTH X1Y 4  Latency: 144  ns
GTH X1Y 5  Latency: 144  ns
$
```

The DSS normal data flow is manually triggered by the user using the TTC. The transfers can be rearmed to be retriggered many times.

If some words with errors are detected, it is desirable that the user performs a BER evaluation using DSS BER data flow, described in subsection 4.3.6. The result of the BER process performed at DS Sender and Receiver from physical slot 1 is:

```
$ python ipbus/BER_allgth.py
Sender Network is 192.168.102.0
Receiver Network is 192.168.101.0
Type the IP Last Byte:
101
Sender Board: 192.168.102.101
Receiver Board:  192.168.101.101

Start BER measurement...
Process takes 7 minutes...
3 min ...
6 min ...
Done!
Reading BER ...

GTH X0Y 0  BER: E- 10
GTH X0Y 1  BER: E- 12
GTH X0Y 2  BER: E- 7
...
GTH X0Y 32  BER: E- 7
GTH X0Y 33  BER: E- 10
GTH X1Y 0  BER: E- 12
GTH X1Y 1  BER: E- 12
...
```

```
25 GTH X1Y 4   BER: E- 12
26 GTH X1Y 5   BER: E- 9
27 $
```

It was possible to check the integrity of all 480 GTH links in the shelf. The results are shown and analyzed in the next section.

Despite all DS pair of boards present the same tune parameters in the FPGA design, still it is possible to change the tuning of each GTH channel without reprograming the board, through IPbus comands. The user input overwrites the hardcoded tuning set in the board. The interface from the python script is illustrated below:

```
1 $ python ipbus/Tuning_Sen.py
2 Sender Network is 192.168.102.0
3 Type the Sender IP Last Byte:
4 101
5 Tuning Board:  192.168.102.101
6 Please Wait ...
7 Set DSB Module Index from 1 to 40: 30
8 0x78000
9 Set TX Post-cursor Index from 1 to 32: 8
10 0b01000
11 Set TX Pre-cursor Index from 1 to 32: 8
12 0b01000
13 Set TX Diff Swing Index from 1 to 16: 8
14 0b1000
15 Tuning    (19 downto  0) set to:  0b01111001000010001000
16 DSB Index (19 downto 14) set to:  0b011110 30th DSM X0Y29 GTH
17 TX Pre    (13 downto  9) set to:  0b01000  1.94 dB
18 TX Post   ( 8 downto  4) set to:  0b01000  1.94 dB
19 TX Diff   ( 3 downto  0) set to:  0b1000   807 mV
20 $
```

All IPbus commands tested in this section are used in the shelf level tests in the next Section.

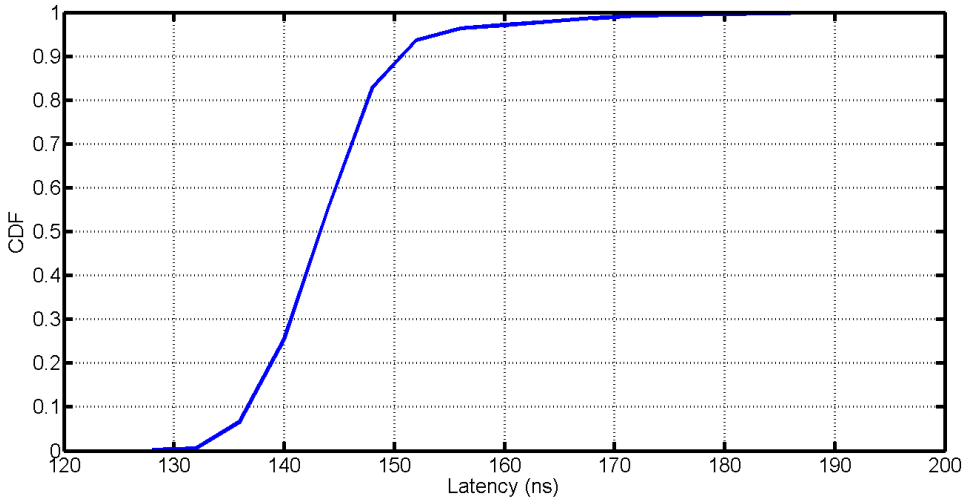
5.2 Data Sourcing Shelf Level Tests

The DS shelf level test, as described in Chapter 3 and illustrated in Figure 3, uses two ATCA shelves with 12 Pulsar 2b boards in each. The bottom ATCA shelf is programmed with the DS Sender, and the upper one with the Receiver, both operating links with 8 Gbps speed rate. The goal in this case is to check the performance of the 120 QSFP+ optical cables, 480 MGT channels, in the RTMs using a specific wrapper communication protocol.

In normal data flow, the data is loaded into the sender BRAMs via IPbus using Python scripts, as described in Section 4.2. A file can be used to load data samples to be transmitted by the 480 MGTs. After loading the data, the user can start the data flow using the sync EC0 signal, as described in Subsection 3.3. All TTC signals are received by one board in each shelf. That board is responsible to receive the TTC signals and replicate them through the backplane.

The first shelf level test performed aimed to measure the latency of all 480 links in the setup. Table 5 shows the latency spread in the channels of the shelf level test. A Cumulative Distribution Function (CDF) of the latency measured ten times is shown in Figure 31.

Figure 31 - CDF Latency shelf level.



Source: Created by the author.

The result shows that more than 90% of the links have latency lower or equal than 150 ns. The distribuition shows that some channels presented latency greater to 170 ns, but always inside a 200 ns latency time window.

The difference between the faster and slower latency of the links, shows that the synchronization time window is roughly 70 ns. This shows that the Data Sourcing was able to measure and evaluated the latency and synchronization of the serial link protocols, at sender and receiver sides, plus optical signal propagation. The cables length in this setup are not uniform and between 3-6 meters. Considering the light propagation speed in the fiber, due to cable length the signal propagation delay can have 15-30 ns.

The latency measurement can be applied to different types of communication media, like copper or even radio frequency.

The reading process at the receiver side was used to detect a couple of errors in the transmission comparing the data received with the original data sent. The BER tests, described in Subsection 4.3.6, were used to evaluate the integrity of the channels in shelf Level.

Table 6 shows the BER spread in the shelf level. In this case, all tuning parameters used the initial values, as described in Section 2.3. Table 7 shows the BER spread in the shelf level when the tuning parameters used the opmitized values found by Auto Tuning System, as described in Section 2.3.

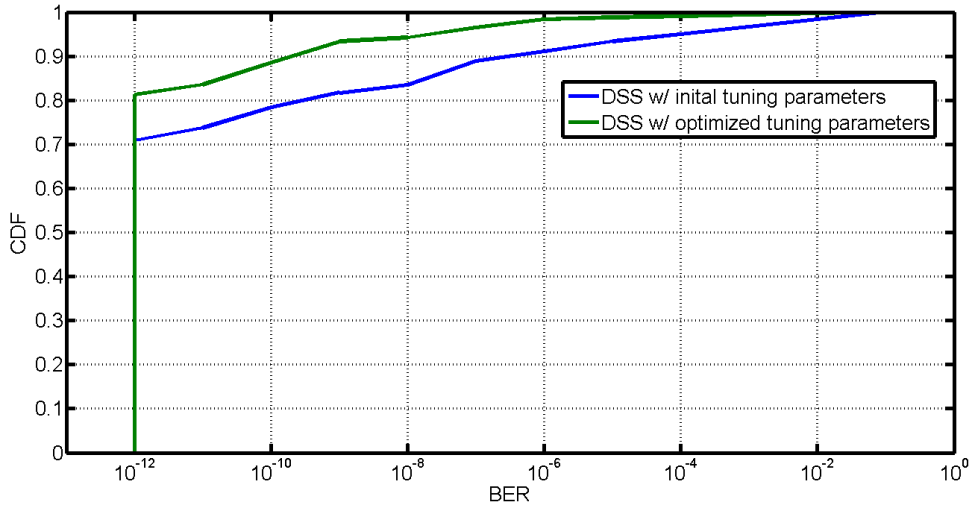
A CDF of the BER, illustrated in Figure 32, shows a comparison between two sets of tuning parameters, as explained in Section 2.3.

Table 5 - Latency shelf level.

Channels\ physlots		Latency (ns)											
		phy1	phy2	phy3	phy4	phy5	phy6	phy9	phy10	phy11	phy12	phy13	phy14
GTH X0Y	0	136	136	140	148	136	140	144	144	136	148	140	144
GTH X0Y	1	140	136	136	144	128	144	144	144	140	148	140	144
GTH X0Y	2	140	140	140	140	140	136	140	144	140	136	136	144
GTH X0Y	3	140	136	140	144	164	136	144	144	140	144	136	144
GTH X0Y	4	144	140	144	140	168	144	144	148	140	148	144	152
GTH X0Y	5	140	140	144	140	168	148	148	148	148	148	148	144
GTH X0Y	6	144	144	144	140	168	144	144	148	152	152	140	144
GTH X0Y	7	140	144	144	136	168	140	148	148	140	144	148	152
GTH X0Y	8	144	144	148	144	168	140	140	148	148	140	152	148
GTH X0Y	9	144	140	144	136	156	140	140	144	148	140	148	148
GTH X0Y	10	148	136	144	144	140	148	144	144	152	140	144	148
GTH X0Y	11	148	140	144	140	144	140	140	144	152	140	148	148
GTH X0Y	12	144	148	144	140	144	144	140	144	144	148	140	144
GTH X0Y	13	148	152	144	140	156	148	140	148	148	148	140	148
GTH X0Y	14	148	148	140	144	188	144	140	156	144	148	144	144
GTH X0Y	15	152	176	144	148	164	144	140	156	144	148	140	148
GTH X0Y	16	148	148	140	140	140	144	140	144	140	140	140	144
GTH X0Y	17	148	180	144	148	136	144	140	148	148	136	140	152
GTH X0Y	18	148	144	144	148	136	148	144	144	148	140	140	156
GTH X0Y	19	144	148	144	148	136	140	140	144	148	140	144	148
GTH X0Y	20	148	152	148	144	132	144	144	144	148	144	152	148
GTH X0Y	21	144	144	152	148	128	148	140	144	144	148	148	152
GTH X0Y	22	148	148	148	144	176	152	140	148	148	144	152	148
GTH X0Y	23	148	144	148	144	136	152	140	144	152	152	152	148
GTH X0Y	24	144	152	140	148	144	148	148	148	144	144	148	148
GTH X0Y	25	144	148	144	152	136	152	152	148	144	144	148	152
GTH X0Y	26	144	144	144	152	144	148	152	148	144	144	144	152
GTH X0Y	27	148	148	140	152	144	152	152	152	144	148	140	148
GTH X0Y	28	152	148	144	148	152	148	148	148	144	148	144	144
GTH X0Y	29	152	148	140	148	156	148	148	144	144	144	140	140
GTH X0Y	30	152	144	144	152	156	152	152	148	148	148	144	144
GTH X0Y	31	152	148	140	148	156	152	152	148	140	152	148	144
GTH X0Y	32	152	148	140	144	160	156	152	148	152	148	148	156
GTH X0Y	33	152	144	144	144	196	156	148	148	148	144	152	152
GTH X1Y	0	148	140	136	136	136	136	140	140	140	148	144	144
GTH X1Y	1	148	140	136	140	136	136	140	140	140	148	140	144
GTH X1Y	2	148	144	136	144	136	140	140	140	140	144	140	144
GTH X1Y	3	148	144	144	140	136	144	136	136	144	148	144	148
GTH X1Y	4	144	140	160	144	148	136	144	136	144	144	144	144
GTH X1Y	5	144	148	140	132	144	136	144	136	144	144	144	140

Source: Created by the author. Note: In red channel latency greater than 170 nano seconds.

Figure 32 - CDF BER Shelf Level.

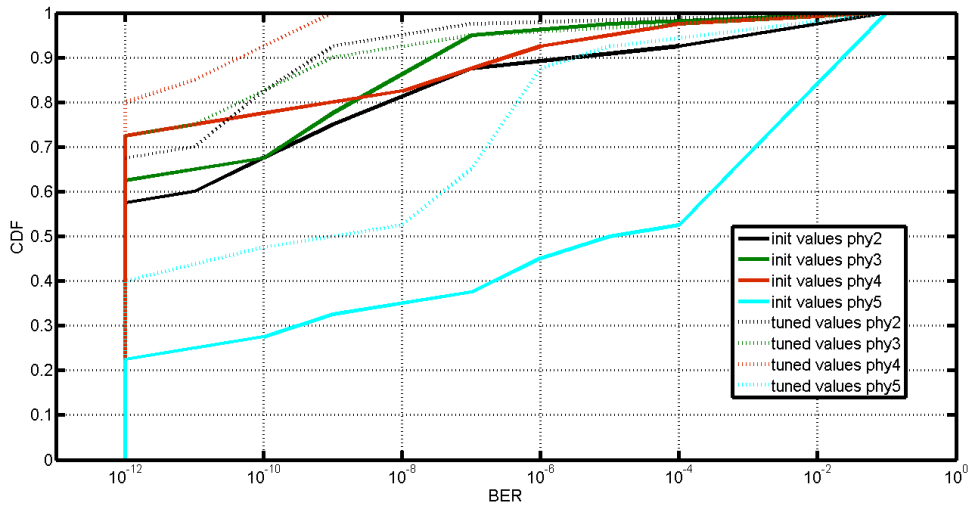


Source: Created by the author.

In the Figure 32 blue curve, 70% of the links have BER lower or equal to 10^{-12}. Due to the tune opmitization process performed by Auto tuning System, the amount of channels with that performance increased to 80% (see the green curve). The BER difference is of almost 4 orders of magnitude for a CDF of 90% between initial and optimized tunning.

The difference of the BER shows that some optimization of the tuning can improve the links performance. Analyzing the tables, the physlots 2 and 3, after the optimized tuning, presents 80% of the channels with BER lower than 10^{-10}. Even physlot 5, that has the worst performance, could improve the integrity by two orders of magnitude. This effect is also illustrated by Figure 33.

Figure 33 - CDF BER separated by physlot.



Source: Created by the author.

Moreover, it is important to say that the setup presents cables from 3 different vendors,

Table 6 - BER shelf level initial tuning.

Channels\ physlots		BER (1E)											
		phy1	phy2	phy3	phy4	phy5	phy6	phy9	phy10	phy11	phy12	phy13	phy14
GTH X0Y	0	-11	-12	-9	-12	-5	-9	-5	-12	-12	-10	-7	-12
GTH X0Y	1	-12	-7	-12	-12	-6	-12	-12	-12	-12	-9	-12	-11
GTH X0Y	2	-5	-12	-7	-4	-1	-5	-6	-8	-5	-6	-7	-9
GTH X0Y	3	-12	-12	-12	-12	-1	-12	-8	-12	-12	-10	-12	-12
GTH X0Y	4	-10	-12	-12	-7	-12	-12	-12	-12	-4	-12	-12	-12
GTH X0Y	5	-12	-4	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	6	-7	-12	-7	-5	-5	-5	-6	-11	-5	-8	-6	-12
GTH X0Y	7	-12	-9	-12	-12	-1	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	8	-12	-12	-7	-10	-1	-12	-8	-12	-9	-9	-10	-12
GTH X0Y	9	-12	-7	-9	-12	-1	-12	-12	-12	-12	-10	-11	-12
GTH X0Y	10	-10	-12	-7	-6	-1	-8	-8	-10	-7	-7	-7	-12
GTH X0Y	11	-12	-7	-12	-12	-6	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	12	-12	-12	-7	-10	-4	-12	-12	-12	-12	-7	-11	-12
GTH X0Y	13	-12	-10	-10	-12	-1	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	14	-12	-1	-12	-12	-1	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	15	-12	-12	-10	-12	-1	-12	-10	-12	-12	-9	-11	-12
GTH X0Y	16	-12	-1	-12	-12	-1	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	17	-12	-12	-12	-12	-1	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	18	-12	-12	-12	-12	-1	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	19	-12	-9	-12	-12	-1	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	20	-12	-7	-12	-12	-1	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	21	-12	-9	-12	-12	-1	-12	-12	-12	-12	-12	-12	-11
GTH X0Y	22	-12	-12	-12	-12	-1	-12	-12	-12	-12	-11	-12	-12
GTH X0Y	23	-12	-12	-12	-12	-1	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	24	-12	-12	-12	-12	-9	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	25	-12	-10	-12	-12	-7	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	26	-12	-12	-9	-12	-7	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	27	-12	-12	-9	-12	-9	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	28	-12	-11	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	29	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	30	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	31	-12	-7	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	32	-12	-10	-7	-7	-1	-12	-11	-12	-12	-12	-10	-12
GTH X0Y	33	-12	-12	-7	-8	-1	-12	-12	-12	-12	-12	-10	-12
GTH X1Y	0	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X1Y	1	-12	-12	-12	-12	-10	-12	-10	-12	-12	-12	-10	-12
GTH X1Y	2	-9	-12	-12	-6	-12	-12	-10	-12	-11	-12	-12	-8
GTH X1Y	3	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X1Y	4	-11	-4	-5	-8	-11	-12	-12	-11	-10	-12	-12	-7
GTH X1Y	5	-4	-12	-1	-1	-6	-4	-5	-7	-4	-6	-7	-12

Source: Created by the author. RED: Channels with BER greater than 10^{-4}. GREEN: Channels with BER greater than 10^{-9}. BLUE: Channels with BER smaller than 10^{-10}.

Table 7 - BER shelf level optimized tuning.

Channels\ physlots		BER (1E)											
		phy1	phy2	phy3	phy4	phy5	phy6	phy9	phy10	phy11	phy12	phy13	phy14
GTH X0Y	0	-12	-10	-10	-12	-6	-12	-7	-12	-11	-10	-11	-12
GTH X0Y	1	-12	-12	-12	-12	-1	-12	-12	-12	-12	-9	-12	-9
GTH X0Y	2	-12	-9	-8	-9	-1	-12	-10	-12	-9	-7	-10	-9
GTH X0Y	3	-12	-12	-12	-12	-6	-12	-9	-12	-12	-9	-12	-12
GTH X0Y	4	-12	-12	-12	-11	-12	-12	-12	-12	-7	-12	-12	-12
GTH X0Y	5	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	6	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	7	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	8	-12	-12	-11	-12	-6	-12	-12	-12	-10	-12	-11	-12
GTH X0Y	9	-12	-12	-12	-12	-6	-12	-12	-12	-12	-11	-12	-12
GTH X0Y	10	-12	-10	-9	-10	-1	-12	-9	-12	-8	-9	-10	-10
GTH X0Y	11	-12	-12	-12	-12	-7	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	12	-12	-10	-10	-12	-6	-12	-12	-12	-12	-11	-12	-12
GTH X0Y	13	-12	-12	-12	-12	-6	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	14	-12	-12	-12	-12	-6	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	15	-12	-7	-12	-12	-6	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	16	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	17	-12	-1	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	18	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	19	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	20	-12	-12	-12	-12	-10	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	21	-12	-7	-9	-11	-5	-12	-12	-12	-12	-10	-12	-10
GTH X0Y	22	-12	-9	-12	-12	-8	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	23	-12	-12	-12	-12	-9	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	24	-12	-12	-12	-12	-7	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	25	-12	-12	-12	-12	-7	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	26	-12	-12	-12	-12	-6	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	27	-12	-9	-9	-12	-7	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	28	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	29	-12	-9	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	30	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	31	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	32	-12	-12	-10	-10	-7	-12	-12	-12	-12	-12	-12	-12
GTH X0Y	33	-12	-10	-7	-10	-5	-12	-12	-12	-12	-12	-11	-12
GTH X1Y	0	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-9
GTH X1Y	1	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X1Y	2	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-9
GTH X1Y	3	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12	-12
GTH X1Y	4	-11	-11	-1	-9	-10	-12	-12	-9	-12	-12	-10	-9
GTH X1Y	5	-12	-10	-1	-9	-10	-12	-12	-12	-10	-12	-12	-8

Source: Created by the author. RED: Channels with BER greater than 10^{-4}. GREEN: Channels with BER greater than 10^{-9}. BLUE: Channels with BER smaller than 10^{-10}.

what may have influenced the BER results. However, a deep study in the hardware can show differences between Pulsar 2b boards due to the manufacture process.

The result shows that the Data Sourcing System was able to measure the BER of the channels, and that it is sensible to tuning processes. According to the application tested, the Data Sourcing is able to measure the BER and calibrate the channels using the "Tune Parameters" command through the IPbus, as described in Subsection 4.4.2.

The DSS was able to prove itself operational. The FNAL ATCA setup performance were evaluated and the best 10 boards were used in the case of study. The next Chapter presents the DS Sender shelf performing a case of study as test system for HEP trigger system development.

6 CASE OF STUDY - AM+FPGA A L1 TRIGGER PROPOSAL

This work is part of an international collaboration. Instituitions in USA, German, France, Italy, China and Brazil joined together to propose a Level 1 Tracking Trigger (L1TT) for CMS Outer Tracker based in Associative Memory (AM) ASIC plus FPGA (AM+FPGA) approach. The effort, led by Fermilab LHC Physics Center (LPC) R&D, uses the same hardware environment already described in this thesis (see Chapter 3).

The goal of the collaboration was to demonstrate the L1TT AM+FPGA approach feasibility by the end of 2016. The demonstration included the hardware implementation and measurements performed at the FNAL ATCA setup.

Before describing the case of study, Section 6.1 stablishes a context of the High Luminosity LHC (HL-LHC). The demonstration goals are described in Section 6.2. The L1TT AM+FPGA approach concepts are described in Section 6.3. Finally, Section 6.4 shows the usage of DSS in integration with the proposed trigger data delivery system.

6.1 LHC Run 3 Upgrade Context

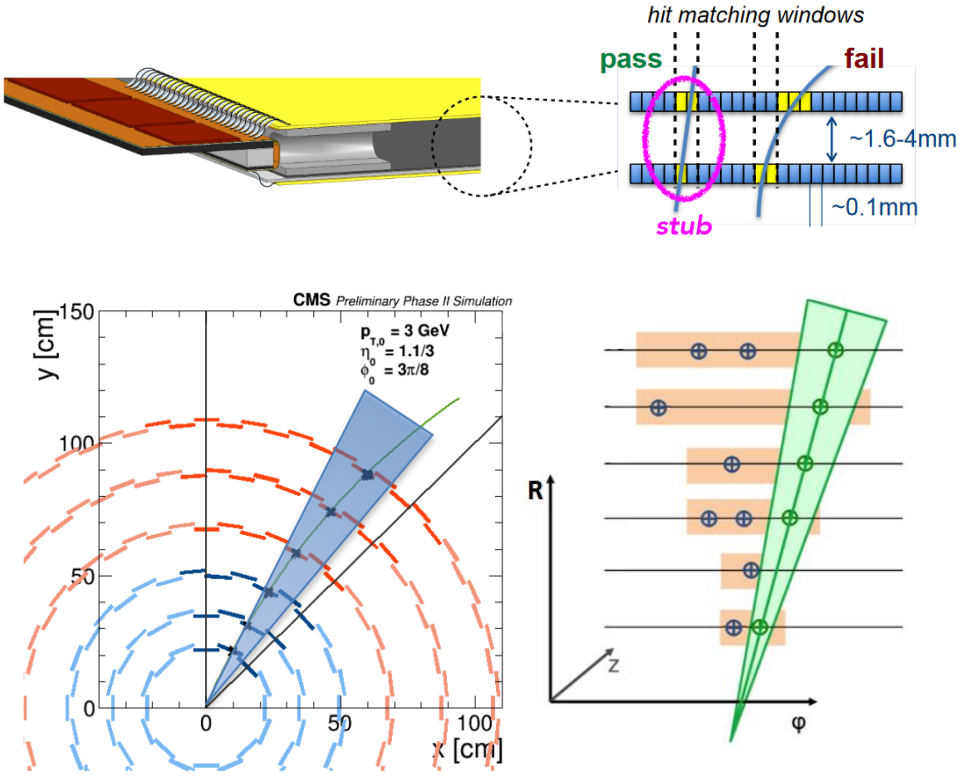
The CMS Technical Proposal of 2015 presents the upgrade foreseen to prepare the CMS experiment for the HL-LHC, also known as Run 3 Upgrade. In the LHC Run 3, the accelerator will increase the amount of simultaneous collisions (pile up), starting in 2022. Each BX will provide 140 pile up, generating a raw data rate of 1 Mbps in a frequency of 40 MHz (CMS COLLABORATION, 2015).

This upgrade will substantially enlarge the mass reach in the search for new particles and extend the potential to study the properties of the Higgs boson discovered at the LHC in 2012. In order to meet the experimental challenges of unprecedented collisions luminosity, the CMS collaboration will need to address the aging of the present detector and to improve the ability of the apparatus to isolate and precisely measure the products of the most interesting collisions. So, the L1 trigger should be able to reduce the original raw data from 40 Tbps to 100 Mbps.

The upgrade of the L1 Trigger will include, for the first time, the Outer Silicon Tracker L1 trigger. The outer tracker front end (FE) electronics are the sensor modules at the detector. Electrically, a sensor module consists of two FE hybrids interfacing to the two rows of strips of the silicon sensor(s) integrating the pixelated sensor with its readout chips. When a charged particle hits both strip layers, in a defined small bending, it generates a pre trigger data named stub, as illustrated in Figure 34 in the top image. The small bending indicates that the stub

belongs to a primary particle track with a predefined minimum momentum. The generic name for the sensor that converts the particle hits in stub data is the CMS Binary Chip (CBC).

Figure 34 - Top - Stub Generation. Bottom - Track formation.



Source: Adapted from (CMS COLLABORATION, 2015).

If the track created by a combination of the 6 outer tracker layers stubs confirms that minimum energy momentum, then it is possible to classify that track as an interesting event, as ilustrated in Figure 34 in the bottom image. This combination and pattern recognition should be performed by the L1TT system.

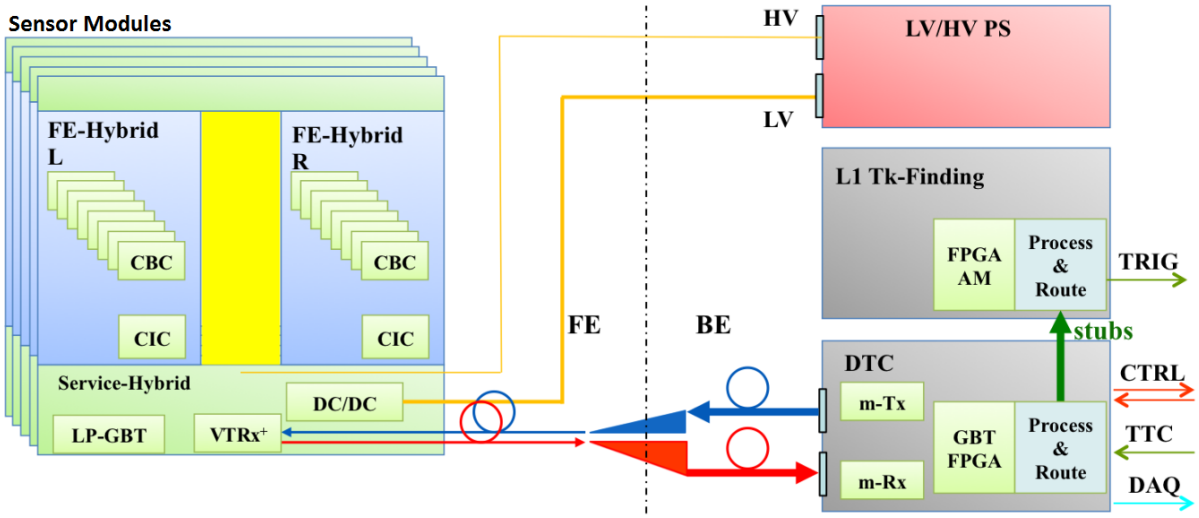
Figure 35 illustrates the data flow between the front-end electronics, back-end electronics, which includes the L1TT system. Data generated by 8 CBCs are buffered, aggregated and formatted by the Concentrator IC (CIC). This service hybrid component serializes the data and sends to the Data Trigger and Control (DTC) back end (BE) system, outside of the detector. Moreover, the subcomponents of the service hybrid are:

Low Power Gigabit Transceiver - LP-GBT is a (de)serializer with a gigabit transceiver protocol developed by CERN (MITRA et al., 2016).

Versatile TransReceiver Plus - VTRx+ is an optical transreceiver (VASEY et al., 2015).

DC/DC converters together with Low/High Voltage Power Supply (LV/HV PS), provides 12 and 800 V to the detector eletronic devices.

Figure 35 - L1 TT Stubs Data Flow.



Source: Adapted from (CMS COLLABORATION, 2015).

At the backend, the DTC will send and receive data to/from multiple sensor modules. This board is yet to be defined and developed.

The proposed DTC requirements are listed below:

- To have multi tranceivers (m-TX) and multi receivers (m-RX). The data rate is not defined yet.
- To have an FPGA able to (de)serialize GBT links, to process the stubs and devilering them to the L1 track finding system
- To support TTC signals. The TTC which provides synchronization to the overall detector eletronics.
- To have communication with the DAQ system. The DAQ system which receives detector raw data through DTC and transfer to the server farms.

Each bunch crossing will produce on the order of 10.000 stubs in the Outer Tracker subdetector. Only about 5 to 10% of those stubs actually belong to an interesting physics events. So, the goal of the L1TT system is to perform pattern recognition to reconstruct those tracks and discard as many as possible of all the other stubs.

The provision of the pre trigger data for the L1 trigger will require an increase in L1 latency for the tracking trigger calculations, and the combination with the calorimeter and muon trigger information and the increased algorithm complexity. The L1 track trigger information is estimated to be available for processing by the trigger system approximately 4 μsec latency after

the interaction occurs. The tracks are then processed to find the primary vertex of charged particle, like from electrons, muons, taus, or any other particles associated with this vertex. Then, they are matched with objects found by the calorimeter trigger and fitted with tracks found by the muon trigger.

All upgrades to be made to the detector electronics in the CMS Phase II are documented in the CMS Technical Design Report (TDR) of 2017 (KLEIN, 2017). The L1TT definitions and specification are performed in this same document using information collected by the L1TT Demonstration Workshop, that happened in December 2016.

6.2 L1 Tracking Trigger Demonstration for CMS TDR

The goal of the L1TT demonstration was to establish the feasibility of L1TT within 4 μs. Three CMS L1TT approaches (AM+FPGA is one of them) were firstly evaluated by simulation and after the performance demonstrated using small-scale prototype hardware implementations. This small-scale system required to represent the trigger for a detector vertical slice. In the AM+FPGA, the detector regions were sliced in 48 vertical slices, also called trigger towers.

Demonstration study used information obtained from several sources. The hardware demonstration is used to evaluate L1TT performance (e.g. tracking efficiency, parameter resolution, rates) for the processing segmentation related to the proposed systems. The hardware demonstrators also provide measurements of the overall latency associated with L1TT. Those measurements and tests used produced data samples. These datasets, which range from simple single particle events to more complex physics topologies (e.g. neutrino + 140 pile up), provide an understanding of the intrinsic performance of L1TT, and of its dependence on pileup/physics conditions. Average pile up conditions of low (0), medium (140) and high (200) were explored.

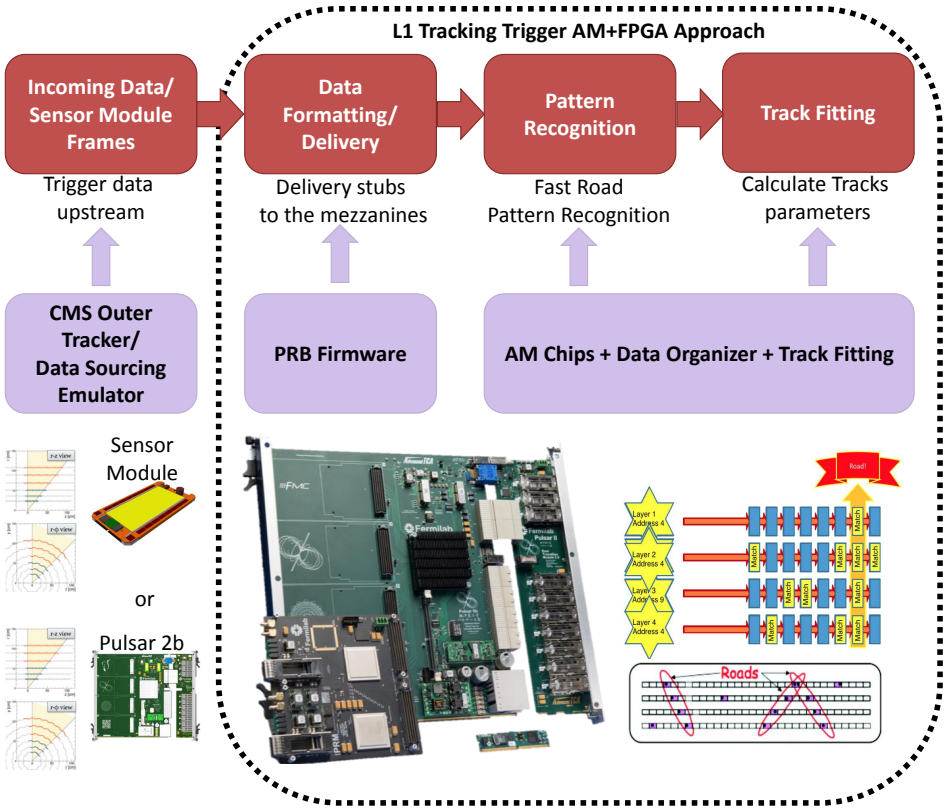
The goal of the hardware implementations was to demonstrate all aspects of L1 track finding and was used to assess the latency incurred by L1 tracking. The demonstrators implemented all salient operations of the proposed systems. Latency was measured from the input of stubs to DTC-equivalent boards, to the output of tracks from the L1TT demonstrator systems.

For the demonstration requirement, the Data Sourcing transfered 512 bits per module (channel), in 200 ns (equivalent to 8 BX data sample). These initial requirement results in a minimum speed of 2.56 Gbps of data rate. However, the module data frames are under development and the current data amount estimation is 1536 bits per 200 ns. Thus, the requirement increased to check if the approach is able to run at least in a data rate of 7.68 Gbps. So, for the L1TT AM+FPGA demonstration, it was defined that latency and BER should be measured in links with this data rate. The hardware testing and results were shown in section 5.2.

6.3 AM+FPGA Approach Concepts

As explained in section 6.1, the pre-trigger data (stubs) is the information used by the L1TT to reconstruct the tracks and decide if those are tracks from an interesting physics events. All L1TT AM+FPGA approach steps are illustrated in Figure 36.

Figure 36 - L1TT AM+FPGA Approach Steps.

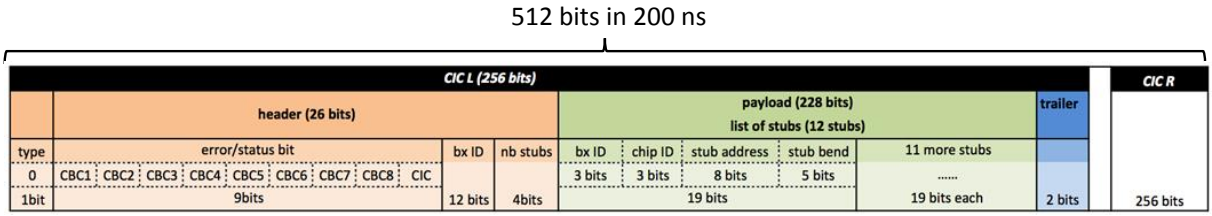


Source: Created by the author.

The input data to the trigger can come from the real detector sensor modules or from an emulator, in this case the DSS described by this work. Therefore, both cases should delivery to L1TT system the sensor module frames. Each frame is composed by a collection of stubs data from left and right CIC related to the latest 8 BX (200 ns). The data format is still under discussion and definitions. The format used for L1TT demonstration has a maximum size of 512 bits, as illustrated in Figure 37, and should arrive to the trigger in a time window of 200 ns.

Each sensor module data is connected with one of the optical channels from the Pulsar 2b RTM and goes directly to the FPGA GTH transceiver. In the AM+FPGA approach, each board in the ATCA shelf would receive data from 40 sensor modules in the same Trigger Tower. In this case, the Pulsar 2b runs the Pattern Recognition Board (PRB) FPGA Design, also known as Data Delivery, developed by the Northwestern University (NWU) partners. The PRBs should exchange stubs between them and format the data arrived and delivery to the mezzanines.

Figure 37 - Demonstration Sensor Modules Data Format.



Source: Adapted from (CMS COLLABORATION, 2015).

The stubs arrive to the mezzanine FPGA, are translated to global coordinates and delivered to the associative memories (AM). The AM ASIC chips compare the coordinates from the predefined pattern bank and return the matched roads. The Pattern Recognition Mezzanine (PRM) FPGA design receives the roads from the AM, performs a track fitting and transfers the track parameter data to the L1 global trigger.

In summary, the case of study focus in the integration between the DSS, more specifically the DS Sender firmware, with the PRB. The work described in the case of study was a joined effort between SPRACE and NWU fellows. The results presented in Chapter 5 evaluated the performance of the FNAL ATCA setup. After that, from the 12 pairs of Pulsars interconnected, we picked the best 10 pairs to perform the case of study. With the bottom ATCA Shelf programmed with the DS Sender firmware, the NWU were users of DSS to finish the PRB development and debug. Section 6.4 describes how the PRB firmware works and the results got from the integration with the DSS.

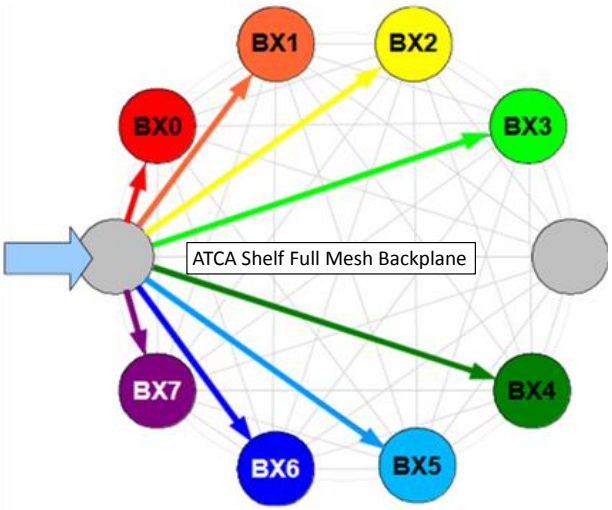
6.4 PRB Shelf Level Integration with Data Sourcing

The PRB or data delivery is a firmware developed, by the NWU fellows, for the Pulsar 2b board. This system receives data from RTM links, 40 links per board, in the format illustrated in Figure 37. The sensor module data frame concentrates stubs from 8 BX in sequence. The first step in PRBs is to decouple the stubs by BX index, to transform the frame in PRB Format (PRBF) 1, and to send them through the ATCA backplane.

The Pattern Recognition Shelf (PRS) performs the L1TT for one Trigger Tower using 10 Pulsar 2b boards. The system rotates funcionalities to have one Pulsar receiving through the backplane the stubs related to the same BX ID from the other nine. Therefore, there is in one proccessing cycle 8 Pulsar 2b receiving stubs from 8 different BX and 2 Pulsar 2b idle, as illustrated in Figure 38. After receiving the stubs from the same BX, PRB sorts them by tracker layer (from 1 to 6) and handles to PRBF 2.

Each Pulsar 2b in PRS hosts two AM mezzanines. The PRBF 2 frames are forwarded to one mezzanine one at a time. The permute happens using Round Robin scheduling. The PRS

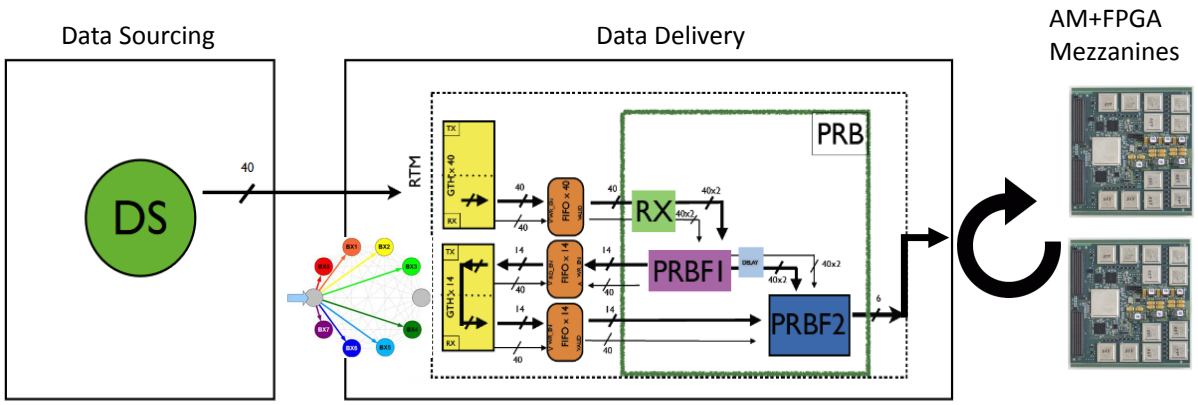
Figure 38 - PRB Backplane data sharing.



Source: Created by the author.

system steps are illustrated and summarized in Figure 39.

Figure 39 - PRB System Summary.

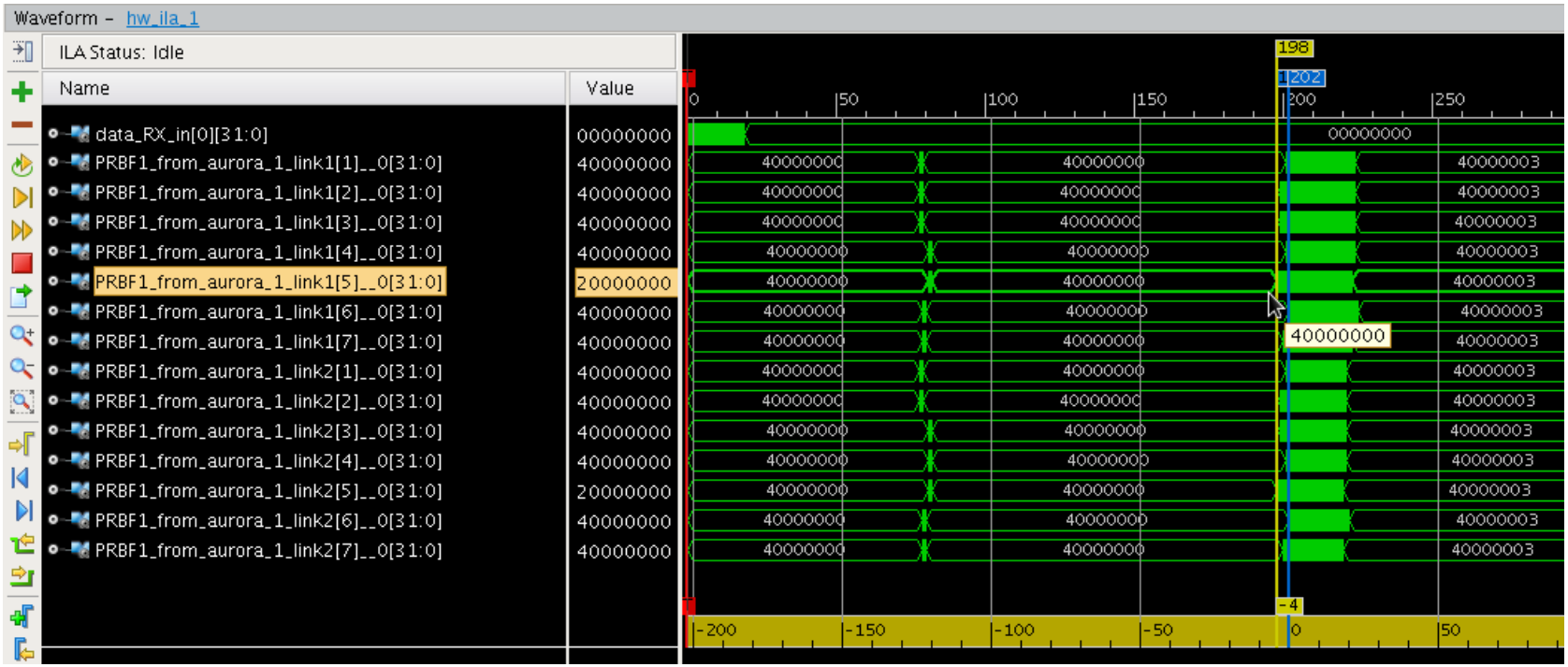


Source: Created by the author.

The BER and latency results got by DS shelf level test allowed the collaboration to start the integration between the detector emulator and the proposed trigger. The first step was to integrate the DS Sender, acting as outer tracker modules emulator (upstream), with the data delivery downstream system.

For the demonstration, PRB should process all stubs alligned in the same time window and check the PRBF2 with the PRB simulator software. Figure 40 illustrates all stubs from a shelf level integration between Data Sourcing and Pattern Recognition Shelves. The result shows that the stubs are aligned with the 200 ns time window. The latency jitter analysis was discussed in Section 5.2.

Figure 40 - PRBF received data.



Source: Created by the author.

From what concerns to the latency in the DS Sender to PRB scenario, test points from the FPGAs were connected an oscilloscope to the latency step flags described as follows:

T0 - Test point at DS It marks the first 32 bit word sent out to the wrapper scrambler in TX side. This flag can emulate data starts to upstream from the detector.

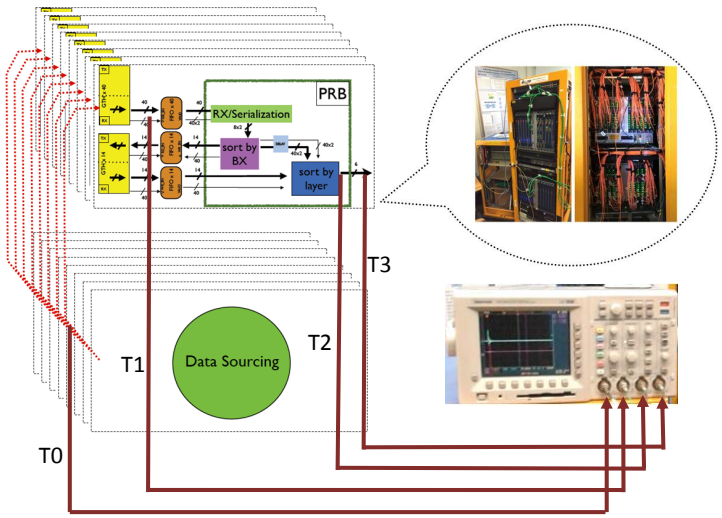
T1 - 1st Test point at PRB This flag rises when the first word arrives to any PRB channel.

T2 - 2nd Test point at PRB It rises when the first stub in PRBF2 is ready to be sent to the mezzanines.

T3 - 3rd Test point at PRB It rises when the last stub in PRBF2 is ready to be sent to the mezzanines.

Figure 41 illustrates the testpoint diagram connection to the demonstration setup.

Figure 41 - Oscilloscope latency measurement scheme.



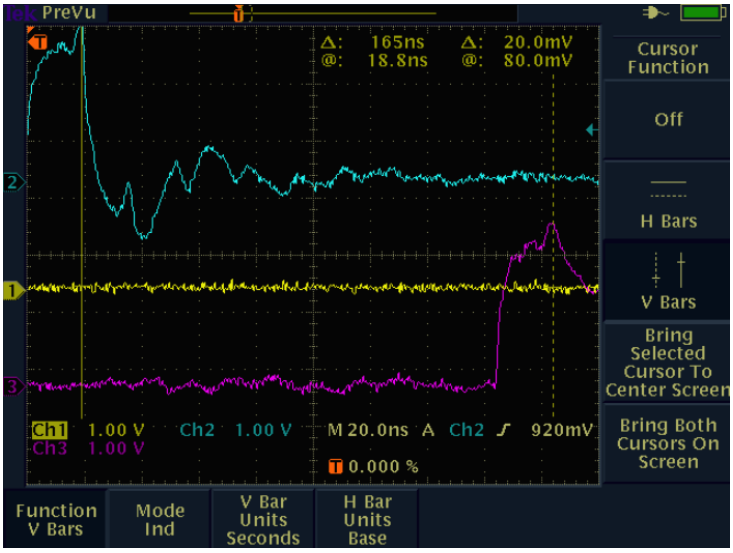
Source: Created by the author.

From the latencies measured with the oscilloscope, this work just analyzes the T0 to T1 flag signals. This interval shown in Figure 42 validates the internal latency measurement of the Data Sourcing Receiver. In both systems, the latency for 8 Gbps are inside a 200 ns time window, as required for the demonstration.

The latency considered between the T2 and T3 steps is related to PRB processing steps, and does not concerns about the DS Sender results.

With those results presented in this work, all requirements for the Data Sourcing were reached for the L1TT AM+FPGA hardware demonstration. However, data sourcing system can be applied for other applications and test any upstream or downstream system.

Figure 42 - Oscilloscope latency measurement from T0 in light blue and T1 in magenta.



Source: Created by the author.

In an upstream system, it is possible to connect another hardware with the Data Sourcing Receiver firmware programmed. Then BER, latency and data format can be evaluated by the electronic board and the user interface. In other hand the Data Sourcing Sender can emulate an upstream, like the detector modules in this work, and provide data in high data rates.

The features of the DSS are stable, but there is room for improvements in tuning and link analysis.

7 CONCLUSIONS AND OUTLOOK

Thanks to the newer FPGA families, DAQ, or data formatters, have been developed in different areas using those digital systems. Those systems require high number of I/Os, synchronism, and processing. The Data Sourcing System (DSS) can be used to test any up or downstream system and to help the debugging process. This system is able to calibrate the tuning parameters of the channels measuring BER and latency.

As a case of study, the DSS was used as part of the AM+FPGA proposal of the CMS Outer Tracker L1 trigger. The emulation performed a mimic of the detector incoming data resultant from the collision events. The system was aimed to help the collaboration to debug and demonstrate the proposed trigger feasibility. The NWU fellows, developers of the PRB firmware, were the main users from DSS.

Despite being developed for the proposed CMS L1 trigger demonstration, the DSS has presented features that make the system to be for general purposes. There are 131 kbits available for each channel to be written or read by the user. Considering the shelf level scenario, the DSS is able to transfer more than 62 Mbits with top speed of 3.84 Tbps. The related work shown that never such a bandwidth were achieved in a system level testing and operations.

To increase the amount of data to be transfered, more memory blocks can be used in DSM. The FPGA utilization leaves room to increase the BRAM usage by four times at least.

The sync system using TTC ensures that the latency of all 480 links is within a time windows of 200 ns. Moreover, 90% of those links have latency less than 150 ns. The signals propagated through the backplanes of different ATCA shelves, showing that a similar behaviour can be expected for a multiple shelves scenario. A simple reset can be performed with the user interface to rearm the system. The Data Sourcing also performs some other functions, such as latency measurements, BER measurements and tuning parameter settings.

As shown in this thesis, the BER measurement is sensible to tuning optimization. The parameters of each channel can be changed without reprograming the FPGA.

The DSS test and results, together with the case of study, shown that this system is promissing to help the commissioning, development, debugging and testing of an upstream or downstream system. The remote access allows that those tests be performed in system of difficult physical access and performed by world wide collaborations, as collaboration of CMS or LHC have.

7.1 Future Work

The DSS can be used to provide deeper studies of the links performance. Using the BER and latency measurements it is possible to continue increasing the data rate of the serial protocols and debug the decreasing of performance.

It is possible also to combine the DS Sender and the DS Receiver FPGA Designs into a single design to support single shelf loopback. Or even to add MGT control registers to IPBUS to allow eye scan measurements while DS is running.

The GT Wizard for Aurora 64b/66b protocol was used by DSS in 8 Gbps. Other protocols can be developed or used to increase the links performance. The wrapper interface made DSS interoperable between different high speed serial links.

The DSS FPGA designs are focused in Xilinx FPGAs. Thus, changing the PCB hardware with those this FPGA vendor, still possible to integrate DSS with minor changes as number of links available or even a different clock system employed. It is planned to use DSS as an emulator of another CMS L1TT approach that uses μTCA standard. So, DSS will be adapted to operate in different hardware setup with other requirements.

The DSS can evolve to a commissioning system. Features like XVC, IPMI, IPbus, if concentrated in a web system, could allow even untrained people to monitor the hardware behaviour and results from different sites of a collaboration.

7.2 Publications

The current work already published part of the results in the following:

- L. A. Ramalho, A. M. Cascadan, A. A. Shinoda, V. Finotti, J. Olsen, T. T. Liu, and Z. Hu, A FPGA Based Data Sourcing Emulator and Multiple Channel System, at Journal of Instrumentation - JINST (2017). - SUBMITTED in 07/11/2017.

- L. A. Ramalho, A. A. Shinoda, V. F. Ferreira, A. M. Cascadan, A 320 Gbps board level Data Sourcing System FPGA Design, Oral presentation at Conferência Brasileira de Dinâmica, Controle e Aplicações (DINCON), São José do Rio Preto (SP), Brazil (2017). - PUBLISHED AT http://www.dincon2017.com.br .

- T. C. Paiva, L. A. Ramalho, A. A. Shinoda, A. Cascadan, V. F. Ferreira, and M. Vaz, "A Framework for Development and Test of XTCA Modules with FPGA Based Systems For Particle Detectors", PCaPAC - 11th International Workshop Personal Computers and Particle Accelerator Controls., São Paulo, Brazil (2016). - PUBLISHED AT http://accelconf.web.cern.ch/AccelConf/pcapac2016/papers/thdaplco06.pdf.

- S. Ajuha, A. Cascadan, T. C. de Paiva, S. Das, R. Eusebi, V. F. Ferreira, K. Hahn, Z. Hu, S. Jidariani, J. Konigsberg, T. T. Liu, J. F. Low, Y. Okumura, J. Olsen, L. A. Ramalho, R. Rossin, L. Ristori, A. A. Shinoda, N. Tran, M. Trovato, K. Ulmer, M. Vaz, X. Wen, J. Wu, Z. Xu, H. Yin, and S. Zorzetti, "A Full Mesh ATCA-based General Purpose Data Processing Board (Pulsar II) Fermilab Technical Memo TM-2650-E", Fermilab, Batavia IL, United States (2017).

- CMS Collaboration - "Technical Design Report - The Phase-2 Upgrade of the CMS Tracker", CERN, Geneva, Switzerland (2017). (KLEIN, 2017)

REFERENCES

ACOSTA, D. CMS Trigger Improvements Towards Run II. **Nuclear and Particle Physics Proceedings**, Amsterdam, v. 273-275, n. 1, p. 1008–1013, 2016.

ALENA, R.; COLLIER, P.; AHKTER, M.; SINHAROY, S.; SHANKAR, D. High Performance Space VPX Payload Computing Architecture Study. **IEEE Aerospace Conference**, Piscataway, v. 1, n. 1, p. 1–19, 2016.

ALOISO, A.; GIORDANO, R.; IZZO, V.; PERRELLA, S. A Frequency Agile, Self-Adaptive Serial Link on Xilinx FPGAs. **IEEE Transactions on Nuclear Science**, Piscataway, v. 62, n. 3, p. 955–962, June 2016.

BARRIENTOS, D.; GONZALEZ, V.; BELLATO, M.; GADEA, A.; BAZZACCO, D.; BLASCO, J. M.; BORTOLATO, D.; EGEA, F. J.; ISOCRATE, R.; PULLIA, A.; RAMPAZZO, G.; SANCHIS, E.; TRIOSSI, A. Development of the Control Card for the Digitizers of the Second Generation Electronics of AGATA. In: IEEE-NPSS Real Time Conference, 18, 2012, Berkeley. **Proceedings of the...** Berkeley: IEEE, 2012. Disponível em: <http://ieeexplore.ieee.org/document/6418205/>. Acesso em: 11 maio2017.

BATISTA, A. J. N.; NETO, A.; CORREIA, M.; FERNANDES, A. M.; CARVALHO, B. B.; FORTUNATO, J. C.; SOUSA, J.; VARANDAS, C. A. F.; SARTORI, F.; JENNISON, M. ATCA Control System Hardware for the Plasma Vertical Stabilization in the JET Tokamak. In: IEEE-NPSS Real Time Conference, 16, 2009, Beijing. **Proceedings of the ...** Beijing: IEEE, 2009. Disponível em: <http://ieeexplore.ieee.org/document/5321967/>. Acesso em: 11 maio 2017.

BAUSS, B.; BROGNA, A.; BUCHER, V.; DEGELE, R.; KAHRA, H.; RAVE, S.; ROCCO, E.; SCHAFER, U.; SOUZA, J.; TAPPROGGE, S.; WEIRICH, M. Latest Frontier Technology and Design of the ATLAS Calorimeter Trigger Board Dedicated to Jet Identification for the LHC Run 3. In: IEEE Nuclear Science Symposium and Medical Imaging Conference, 2016, Strasbourg. **Proceedings of the...** Strasbourg: IEEE, 2016. Disponível em: <http://ieeexplore.ieee.org/document/8069839/>. Acesso em: 11 maio 2017.

BELLATO, M.; BORTOLATO, D.; CHAVAS, J.; ISOCRATE, R.; RAMPAZZO, G. Sub-nanosecond clock synchronization and trigger management in the nuclear physics experiment AGATA. **Journal of Instrumentation**, Bristol, v. 8, n. 1, p. 1–19, 2013.

BROOKE, J. Performance of the CMS Level-1 Trigger. **36th International Conference on High Energy Physics**, v. 177, n. 1-2, p. 1–6, 2012. Disponível em: <https://arxiv.org/pdf/1302.2469v1.pdf>. Acesso em: 11 maio 2017.

BUHROW, B. R.; GOETZINGER, W. J.; GILBERT, B. K. 1 Tb/s Anti-Replay Protection with 20-port On-Chip RAM Memory in FPGAs. IEEE International Conference on ReConFigurable Computing and FPGAs (ReConFig), 2016, Cancun. **Proceedings**

of the... Cancun: IEEE, 2016., v. 1, n. 1, p. 1–8, 2016. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7857190>. Acesso em: 11 maio 2017.

BUNKOWSKI, K.; POZNIAK, K. T.; BLUJ, M.; DOROBA, K.; ISKANIUS, M.; KALINOWSKI, A.; KIERKOWSKI, K.; KONECKI, M.; KROLIKOWSKI, J.; KUDLA, I.; LODDO, F.; OKLINSKI, W.; WROCHNA, G.; ZABOLOTNY, W. Synchronization methods for the PAC RPC trigger system in the CMS experiment. **Measurement Science and Technology**, Bristol, v. 18, n. 8, p. 2446–2455, 2007.

CMS COLLABORATION. **The CMS experiment at the CERN LHC**. 2008. Disponível em: <http://iopscience.iop.org/1748-0221/3/08/S08004>. Acesso em: 11 mai. 2017.

CMS COLLABORATION. **XDAQ CMS Online Software project page**. 2009. Disponível em: <https://svnweb.cern.ch/trac/cmsos>. Acesso em: 11 mai. 2017.

CMS COLLABORATION. Techninal Proposal for the Phase-II Upgrade of the Compact Muon Solenoid. 2015. Disponível em: <https://cds.cern.ch/record/2020886/files/LHCC-P-008.pdf>. Acesso em: 11 mai. 2017.

DAMLE, P.; MATHUR, G.; SEN, A.; KUMAR, T.; MALIK, R.; BANSAL, P. Generic system for characterization of BER and JTOL of high speed serial links. ANNUAL IEEE INDIA CONFERENCE (INDICON), 2015, New Delhi. **Proceedings of the...** New Delhi: IEEE, 2015. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7443453>. Acesso em: 11 maio 2017.

DEVAKUMAR, H.; PANSE, M. S.; KHANDARE, A.; MAYEKAR, M. Design of Lightning Acquisition And Smart Triggering Using Kintex-7 FPGA And NI cRIO. In: IEEE INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ELECTRONICS INFORMATION COMMUNICATION TECHNOLOGY (RTEICT), 2016, Bangalore. **Proceedings of the...** Bangalore: IEEE, 2016, v. 1, n. 1, p. 1429–1435, 2016. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7808068>. Acesso em: 11 maio 2017.

FERENCI, D.; BERROTH, M. A 100 gigabit measurement system with state of the art FPGA technology for characterization of high speed ADCs and DACs. In: RESEARCH IN MICROELECTRONICS ELECTRONICS, 6, 2015, Berlin. **Proceedings of the...** Berlin: IEEE, 2015., v. 1, n. 1, p. 1–4, 2015. Disponível em: <http://ieeexplore.ieee.org/document/5587144/>. Acesso em: 11 maio 2017.

FRANS, Y.; CAREY, D.; ERETT, M.; AMIR-ASLANZADEH, H.; FANG, W. Y.; TURKER, D.; JOSE, A. P.; BEKELE, A.; IM, J.; UPADHYAYA, P.; WU, Z. D.; HSIEH, H.; SAVOJ, J.; CHANG, K. A 0.5 - 16.3 Gb/s Fully Adaptive Flexible-Reach Transceiver for FPGA in 20 nm CMOS. **IEEE Journal of Solid-State Circuits**, Piscataway, v. 50, n. 8, p. 1932–1944, 2015. ISSN 0018-9200.

FRAZIER, R.; ILES, G.; NEWBOLD, D.; ROSE, A. Software and firmware for controlling CMS trigger and readout hardware via gigabit Ethernet. **Physics Procedia**, Amsterdam, v. 37, n. 1, p. 1892–1899, October 2012.

HIDVEGI, A. **FPGA-based Instrumentation for Advanced Physics Experiments**. Tese (Doutorado) — Stockholms University, Stockholm - Sweden, December 2011. Disponível em:

<https://www.diva-portal.org/smash/get/diva2:457982/FULLTEXT02.pdf>. Acesso em: 11 maio 2017.

HILL, J. **Advancing the Use of FPGA Co-Processors through Platforms and High-Level Design Flows**. 2011. Disponível em: <https://www.xilinx.com/support/documentation/white_papers/wp394_Advancing_Coprocessors.pdf>. Acesso em: 11 mai. 2017.

HOLZNER, A.; CMS. **TTCci User Guide**. 2007. Disponível em: <http://cmsdoc.cern.ch/cms/TRIDAS/ttc/modules/ttcci/>. Acesso em: 11 mai. 2017.

HUDA, A.; ANDRESON, J. H. Leveraging Unused Resources for Energy Optimization of FPGA Interconnect. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, Piscataway, v. 1, n. 99, p. 1–14, 2017.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS - IEEE. **IEEE 1014-1987 - Standard for A Versatile Backplane Bus: VMEbus**. 1. ed. German: [s.n.], 1987.

KLEIN, K. **The Phase-2 Upgrade of the CMS Tracker**. Geneva, Jun 2017. Acesso em: 11 mai. 2017.

LARREA, C. G.; HARDER, K.; NEWBOLD, D.; SANKEY, D.; ROSE, A.; THEA, A.; WILLIAMS, T. **IPbus:** a flexible Ethernet-based control system for xTCA hardware. 2015. Disponível em: <http://iopscience.iop.org/article/10.1088/1748-0221/10/02/C02019/meta>. Acesso em: 11 mai. 2017.

LARSEN, R. S. Recent Progress in Next Generation Platform Standards for Physics Instrumentation and Controls. In: IEEE-NPSS REAL TIME CONFERENCE (RT), 18, 2012, Berkeley. **Proceedings of the...** Berkeley: IEEE, 2012, v. 1, n. 1, p. 1–5, 2012. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6418178>. Acesso em: 11 maio 2017.

LEE, W. L.; CHIN, M. S.; CHOO, W. S.; CHEE, C. K. Predictive Method for Simultaneous Switching Output Jitter of DDR for FPGA. In: IEEE INTERNATIONAL ELECTRONICS MANUFACTURING TECHNOLOGY (IEMT), 37, ELECTRONICS MATERIALS AND PACKAGING (EMAP) CONFERENCE, 18, Geoge Tow. **Proceedings of the...** Geoge Tow: IEEE, 2016., v. 37, n. 1, p. 1–5, 2016. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7761972>. Acesso em: 11 maio 2017.

LEMKE, F.; SLOGNAR, D.; BURKHARDT, N.; BRUENING, U. A Unified Interconnection Network with Precise Time Synchronization for the CBM DAQ-System. In: IEEE-NPSS REAL TIME CONFERENCE, 16, 2009, Beijing. **Proceedings of the...** Beijing: IEEE, 2009., p. 506–511, 2009. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5321841>. Acesso em: 11 maio 2017.

LIU, W.; REN, Y.; HE, D.; QIAO, L. Development of universal FMC carrier card based on PXI Express. IEEE ANNUAL UBIQUITOUS COMPUTING AND ELECTRONICS MOBILE COMMUNICATION CONFERENCE (UEMCON), 7, 2016, New York. **Proceedings of the...** New York: IEEE, 2016., 2016. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7777800>. Acesso em: 11 maio 2017.

LLOYD, S.; GOKHALE, M. **Evaluating the feasibility of storage class memory as main**

memory. 2016. Disponível em: <https://e-reports-ext.llnl.gov/pdf/813685.pdf>. Acesso em: 11 maio 2017.

MAKOWSKI, D.; MIELCZAREK, A.; PEREK, P.; NAPIERALSKI, A.; BUTKOWSKI, L.; BRANLARD, J.; FENNER, M.; SCHLARB, H.; YANG, B. High-Speed Data Processing Module for LLRF. **IEEE Transactions on Nuclear Science**, Piscataway, v. 62, n. 3, p. 1083–1090, 2015. ISSN 0018-9499.

MANS, J. **IPBus v1.3:** Simple IP-based ?TCA Control System. 2009. Disponível em: <https://projects.hepforge.org/cactus/trac/wiki/IPbusIntro>. Acesso em: 11 mai. 2017.

MARIN, M. B.; BARON, S.; FEGER, S. S.; LEITAO, P.; LUPU, E. S.; SOOS, C.; VICHOUDIS, P.; WYLLIE, K. The GBT-FPGA core: features and challenges. **Journal of Instrumentation**, Bristol, v. 10, n. 3, p. C03021, 2015.

MCVITTIE, P.; CONTARATO, D.; DENES, P.; DOERING, D.; JOESEPH, J.; WEIZEORICK, J. A Readout System for High-Speed CCD Cameras Based on Advanced Telecommunications Computing Architecture. IEEE NUCLEAR SCIENCE SYMPOSION AND MEDICAL IMAGING CONFERENCE RECORD (NSS/MIC), Anaheim. **Proceedings of the...** Anaheim: IEEE, 2012., v. 63, n. 3, p. 1623–1625, 2012. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6551386>. Acesso em: 11 maio 2017.

MITRA, J.; KHAN, S. A.; MARIN, M. B.; CACHEMICHE, J.; DAVID, E.; HACHON, F.; RETHORE, F.; KISS, T.; BARON, S.; KLUGE, A.; NAYAK, T. K. GBT link testing and performance measurement on PCIe40 and AMC40 custom design FPGA boards. **Journal of Instrumentation**, Bristol, v. 11, n. 2, p. C03039, 2016.

NGUYEN, T. K.; DEKNEUVEL, E.; JACQUEMOD, G.; NICOLLE, B.; ZAMMIT, O.; NGUYEN, V. C. Development of a real-time non-intrusive appliance load monitoring system: An application level model. **International Journal of Electrical Power & Energy Systems**, London, v. 90, n. 1, p. 168–180, 2017.

OHWR. **Optical Clock & Data Recovery FMC**. 2011. Disponível em: <http://www.ohwr.org/projects/optical-cdr-fmc/wiki>. Acesso em: 11 mai. 2017.

OLSEN, J.; LIU, T.; OKUMURA, Y. A full mesh atca-based general purpose data processing board. ***Journal of Instrumentation***, v. 9, n. 1, p. 1–10, January 2014.

PAIVA, T. C. **Remote Development Environment with Reconfigurable Components in the Advanced Telecommuniations and Computing Architecture Context.** 2016. 126f. Disserta??o (Mestrado) — Faculdade de Engenharia, Universidade Estadual Paulista, Ilha Solteira, São Paulo, 2016. Disponível em: <http://repositorio.unesp.br/handle/11449/144448>. Acesso em: 11 maio 2017.

PATEL, R.; SINGH, V. P. A Literature Survey of FPGA Implementation of Inverter Techniques. **International Research Journal of Technology And Applied Science**, Calabar, v. 1, n. 1, p. 1–10, 2017. Disponível em: <http://irjtas.com/uploads/ijm_v01i01n0003.pdf>. Acesso em: 11 maio 2017.

PEREK, P.; MAKOWSKI, D. **Intelligent Platform Management Controller for ATCA Carrier Boards**. Saarbrucken: Lambert Academic Publishing, 2011.

PICMG. **PICMG 3.0 Revision 3.0 AdvancedTCA Base Specification**, [S.l.]. [S.l.], 2008.

RAMALHO, L. A.; PAIVA, T. C.; IOPE, R.; LEAL, B. C.; LIU, T. T.; OLSEN, J.; SHINODA, A. A.; VAZ, M. Development of an Intelligent Platform Management Controller for the Pulsar IIb. In: IEEE NUCLEAR SCIENCE SYMPOSIUM AND MEDICAL IMAGING CONFERENCE (NSS/MIC), 2015, San Diego. **Proceedings of the...** San Diego: IEEE, 2015., 2015. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7581788>. Acesso em: 11 maio 2017.

SARMAH, M. J.; AZEEMUDDIN, S. A Circuit to Eliminate Serial Skew in High-Speed Serial Communication Channels. **IEEE Transactions on Circuits and Systems II: Express Briefs**, Piscataway, v. 62, n. 12, p. 1179–1183, 2015. ISSN 1549-7747.

SPRACE. **Autotuning System for Xilinx MGTs**. 2017. Disponível em: <https://github.com/SPRACE/MGT-Autotuning/>. Acesso em: 11 mai. 2017.

TAUROK, A.; BERGAUER, H.; PADRTA, M. Implementation and synchronisation of the First Level Global Trigger for the CMS experiment at LHC. **Nuclear Instruments and Methods in Physics Research**, Amsterdam, v. 473, n. 3, p. 243–259, May 2001.

TAYLOR, B. G. TTC Distribution for LHC Detectors. **Nuclear Instruments and Methods in Physics Research**, Amsterdam, v. 45, n. 1, p. 821–828, 1998.

TOMEI, T. R. F. P. **Busca por Dimensões Extras no Detector CMS do Large Hadron Collider**. Tese (Doutorado) — Instituto de Física Teórica, Universidade Estadual Paulista, São Paulo, São Paulo - Brazil, 2012. Disponível em: <http://base.repositorio.unesp.br/handle/11449/102537>. Acesso em: 11 maio 2017.

VALDERRAMA, C.; JOJCZYK, L.; POSSA, P. D.; GAZZANO, J. D. FPGA and ASIC convergence. SOUTHERN CONFERENCE ON PROGRAMMABLE LOGIC (SPL), 7, 2011, Cordoba. **Proceedings of the...** Cordoba: IEEE, 2011., v. 1, n. 1, p. 269–274, April 2011. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5782660>. Acesso em: 11 maio 2017.

VASEY, F.; TROSKA, J.; RICCI, D.; MACHADO, S.; PROSSER, A.; HUFFMAN, T.; WEIDBERG, T.; YE, J.; GUI, P. **Versitle Link PLUS Project v2.3**. 2015. Disponível em: <https://espace.cern.ch/project-Versatile-Link-Plus/Shared%20Documents/Versatile%20Link%20PLUS%20Project%20V2.3.pdf>. Acesso em: 11 mai. 2017.

XILINX. **White Paper 419** - Equalization for High-Speed Serial Interfaces in Xilinx 7 Series FPGA Transceivers. 2012. Disponível em: <https://www.xilinx.com/support/documentation/white_papers/wp419-7Series-XCVR-Equalization.pdf>. Acesso em: 11 mai. 2017.

XILINX. **User Guide 768** - Xilinx 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide for HDL Designs. 2013. Disponível em: <https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/7series_hdl.pdf>. Acesso em: 11 mai. 2017.

XILINX. **1G/2.5G Ethernet PCS/PMA or SGMII v15.0 LogiCORE IP Product Guide**. 2015. Disponível em: <https://www.xilinx.com/support/documentation/ip_documentation/gig_ethernet_pcs_pma/v15_0/pg047-gig-eth-pcs-pma.pdf>. Acesso em: 11 mai. 2017.

XILINX. **7 Series FPGAs GTX/GTH Transceivers User Guide**. 2015. Disponível em: <https://www.xilinx.com/support/documentation/user_guides/ug476_7Series_Transceivers.pdf>. Acesso em: 11 mai. 2017.

XILINX. **Aurora 64B/66B v11.0 LogiCORE IP Product Guide**. 2015. Disponível em: <https://www.xilinx.com/support/documentation/ip_documentation/aurora_64b66b/v11_0/pg074-aurora-64b66b.pdf>. Acesso em: 11 mai. 2017.

XILINX. **Virtex-5 Family Overview**. 2015. Disponível em: <https://www.xilinx.com/support/documentation/data_sheets/ds100.pdf>. Acesso em: 11 mai. 2017.

XILINX. **Virtex-6 Family Overview**. 2015. Disponível em: <https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf>. Acesso em: 11 mai. 2017.

XILINX. **Integrated Bit Error Ratio Tester 7 Series GTH Transceivers v3.0**. 2016. Disponível em: <https://www.xilinx.com/support/documentation/ip_documentation/ibert_7series_gth/v3_0/pg152-ibert-7series-gth.pdf>. Acesso em: 11 mai. 2017.

XILINX. **UltraScale FPGA Product Tables and Product Selection Guide**. 2016. Disponível em: <https://www.xilinx.com/support/documentation/selection-guides/ultrascale-fpga-product-selection-guide.pdf>. Acesso em: 11 mai. 2017.

XILINX. **All Programmable 7 Series Product Selection Guide**. 2017. Disponível em: <https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf>. Acesso em: 11 mai. 2017.

XILINX. **UltraScale Plus FPGA Product Tables and Product Selection Guide**. 2017. Disponível em: <https://www.xilinx.com/support/documentation/selection-guides/ultrascale-plus-fpga-product-selection-guide.pdf>. Acesso em: 11 mai. 2017.

XIONG, C.; ZHONG, Y.; ZHANG, C.; YAN, Z. An FPGA Emulation Platform for Polar Codes. In: IEEE INTERNATIONAL WORKSHOP ON SIGNAL PROCESSING SYSTEM (SiPS), 2016, Dallas. **Proceedings of the...** Dallas: IEEE, 2016., p. 148–153, 2016. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7780088>. Acesso em: 11 maio 2017.

ZHANG, L.; ZHANG, K.; CHANG, T. S.; LAFRUIT, G.; KUZMANOV, G. K.; VERKEST, D. Real-time high-definition stereo matching on FPGA. PROCEEDINGS OF THE 19TH ACM/SIGDA INTERNATIONAL SYMPOSIUM ON FIELD PROGRAMMABLE GATE ARRAYS, 19, 2011, Monterey. **Proceedings of the...** Monterey: IEEE, 2011., v. 1, n. 1, p. 55–64, 2011. Disponível em: <https://dl.acm.org/citation.cfm?doid=1950413.1950428>. Acesso em: 11 maio 2017.

ZOU, D. **FPGA-BASED LDPC CODED MODULATIONS FOR OPTICAL TRANSPORT NETWORKS**. Tese (Doutorado) — DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING - University of Arizona, Arizona - USA, 2017. Disponível em: <http://hdl.handle.net/10150/622985>. Acesso em: 11 maio 2017.

APPENDIX A - DATA SOURCING SYSTEM VHDL FILES

The VHDL files from the Data Sourcing System (DSS) are available at a git repository https://git.ncc.unesp.br/laramalho/data-sourcing-system-pub .

- Common files - Composed by TTC, IPbus and any other component common for the DS Sender and Receiver firmwares. TTC and IPbus components were described by the Collaboration that we participated. - Path:./common/*
- DS Sender files - Composed by the Top file, constraints, DSB, DSM, Remote Control, FIFO manager, Tunning parameters components, which were described by this author. - Path:./sender/*
- DS Receiver files - Composed by the Top file, constraints, DSB, DSM, Remote Control, FIFO manager, Tunning parameters, BER and Latency measurements components, which were described by this author. - Path:./receiver/*

APPENDIX B - IPBUS SCRIPTS

In this appendix it is inserted samples of the python scripts written to run IPbus and communicate with Pulsar 2b at FNAL ATCA infrastucture.

- ipbus/sender_connection.xml
- ipbus/receiver_connection.xml
- ipbus/Change_IP_Sen.py
- ipbus/Change_IP_Rec.py
- ipbus/WriteDummy_Sender.py
- ipbus/ReadDummy_Sender.py
- ipbus/Latency_allgth.py
- ipbus/BER_allgth.py
- ipbus/Tuning_Sen.py

Sender Address Mapping XML File.

```
$ cat ipbus/sender_connection.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<node>
    <node id="CMD" address="0x0000" permission="rw" tags="test">
        <node id="IDDLE" address="0x000" mode="single" />
        <node id="NEWIP" address="0x001" mode="single" />
        <node id="RESET" address="0x002" mode="single" />
        <node id="REARM" address="0x003" mode="single" />
        <node id="TUNING" address="0x004" mode="single" />
        <node id="DSBMODE" address="0x005" mode="single" />
        <node id="COUNTER" address="0x00F" mode="single" />
    </node>
    <node id="CHANN0" address="0x1000" permission="rw" mode="block" size="4096" />
    <node id="CHANN1" address="0x2000" permission="rw" mode="block" size="4096" />
    <node id="CHANN2" address="0x3000" permission="rw" mode="block" size="4096" />
    <node id="CHANN3" address="0x4000" permission="rw" mode="block" size="4096" />
    <node id="CHANN4" address="0x5000" permission="rw" mode="block" size="4096" />
    <node id="CHANN5" address="0x6000" permission="rw" mode="block" size="4096" />
    <node id="CHANN6" address="0x7000" permission="rw" mode="block" size="4096" />
    <node id="CHANN7" address="0x8000" permission="rw" mode="block" size="4096" />
    <node id="CHANN8" address="0x9000" permission="rw" mode="block" size="4096" />
    <node id="CHANN9" address="0xA000" permission="rw" mode="block" size="4096" />
    <node id="CHANN10" address="0xB000" permission="rw" mode="block" size="4096" />
    <node id="CHANN11" address="0xC000" permission="rw" mode="block" size="4096" />
    <node id="CHANN12" address="0xD000" permission="rw" mode="block" size="4096" />
    <node id="CHANN13" address="0xE000" permission="rw" mode="block" size="4096" />
    <node id="CHANN14" address="0xF000" permission="rw" mode="block" size="4096" />
    <node id="CHANN15" address="0x10000" permission="rw" mode="block" size="4096" />
    <node id="CHANN16" address="0x11000" permission="rw" mode="block" size="4096" />
    <node id="CHANN17" address="0x12000" permission="rw" mode="block" size="4096" />
    <node id="CHANN18" address="0x13000" permission="rw" mode="block" size="4096" />
    <node id="CHANN19" address="0x14000" permission="rw" mode="block" size="4096" />
    <node id="CHANN20" address="0x15000" permission="rw" mode="block" size="4096" />
    <node id="CHANN21" address="0x16000" permission="rw" mode="block" size="4096" />
    <node id="CHANN22" address="0x17000" permission="rw" mode="block" size="4096" />
    <node id="CHANN23" address="0x18000" permission="rw" mode="block" size="4096" />
    <node id="CHANN24" address="0x19000" permission="rw" mode="block" size="4096" />
    <node id="CHANN25" address="0x1A000" permission="rw" mode="block" size="4096" />
    <node id="CHANN26" address="0x1B000" permission="rw" mode="block" size="4096" />
    <node id="CHANN27" address="0x1C000" permission="rw" mode="block" size="4096" />
    <node id="CHANN28" address="0x1D000" permission="rw" mode="block" size="4096" />
    <node id="CHANN29" address="0x1E000" permission="rw" mode="block" size="4096" />
    <node id="CHANN30" address="0x1F000" permission="rw" mode="block" size="4096" />
    <node id="CHANN31" address="0x20000" permission="rw" mode="block" size="4096" />
    <node id="CHANN32" address="0x21000" permission="rw" mode="block" size="4096" />
    <node id="CHANN33" address="0x22000" permission="rw" mode="block" size="4096" />
    <node id="CHANN34" address="0x23000" permission="rw" mode="block" size="4096" />
    <node id="CHANN35" address="0x24000" permission="rw" mode="block" size="4096" />
    <node id="CHANN36" address="0x25000" permission="rw" mode="block" size="4096" />
    <node id="CHANN37" address="0x26000" permission="rw" mode="block" size="4096" />
    <node id="CHANN38" address="0x27000" permission="rw" mode="block" size="4096" />
    <node id="CHANN39" address="0x28000" permission="rw" mode="block" size="4096" />
</node>
$
```

Receiver Address Mapping XML File.

```
$ cat ipbus/receiver_connection.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<node>
  <node id="CMD" address="0x0000" permission="rw" >
    <node id="IDDLE" address="0x000" mode="single" />
                <node id="NEWIP" address="0x001" mode="single" />
                <node id="RESET" address="0x002" mode="single" />
                <node id="REARM" address="0x003" mode="single" />
                <node id="TUNING" address="0x004" mode="single" />
                <node id="DSBMODE" address="0x005" mode="single" />
                <node id="COUNTER" address="0x00F" mode="single" />
                <node id="BER" address="0x0101" permission="r" mode="block" size="40">
                <node id="LATENCY" address="0x0201" permission="r" mode="block" size="40">
                </node>
  </node>
    <node id="CHANN0" address="0x1000" permission="rw" mode="block" size="4096" />
    <node id="CHANN1" address="0x2000" permission="rw" mode="block" size="4096" />
    <node id="CHANN2" address="0x3000" permission="rw" mode="block" size="4096" />
    <node id="CHANN3" address="0x4000" permission="rw" mode="block" size="4096" />
    <node id="CHANN4" address="0x5000" permission="rw" mode="block" size="4096" />
    <node id="CHANN5" address="0x6000" permission="rw" mode="block" size="4096" />
    <node id="CHANN6" address="0x7000" permission="rw" mode="block" size="4096" />
    <node id="CHANN7" address="0x8000" permission="rw" mode="block" size="4096" />
    <node id="CHANN8" address="0x9000" permission="rw" mode="block" size="4096" />
    <node id="CHANN9" address="0xA000" permission="rw" mode="block" size="4096" />
    <node id="CHANN10" address="0xB000" permission="rw" mode="block" size="4096" />
    <node id="CHANN11" address="0xC000" permission="rw" mode="block" size="4096" />
    <node id="CHANN12" address="0xD000" permission="rw" mode="block" size="4096" />
    <node id="CHANN13" address="0xE000" permission="rw" mode="block" size="4096" />
    <node id="CHANN14" address="0xF000" permission="rw" mode="block" size="4096" />
    <node id="CHANN15" address="0x10000" permission="rw" mode="block" size="4096" />
    <node id="CHANN16" address="0x11000" permission="rw" mode="block" size="4096" />
    <node id="CHANN17" address="0x12000" permission="rw" mode="block" size="4096" />
    <node id="CHANN18" address="0x13000" permission="rw" mode="block" size="4096" />
    <node id="CHANN19" address="0x14000" permission="rw" mode="block" size="4096" />
    <node id="CHANN20" address="0x15000" permission="rw" mode="block" size="4096" />
    <node id="CHANN21" address="0x16000" permission="rw" mode="block" size="4096" />
    <node id="CHANN22" address="0x17000" permission="rw" mode="block" size="4096" />
    <node id="CHANN23" address="0x18000" permission="rw" mode="block" size="4096" />
    <node id="CHANN24" address="0x19000" permission="rw" mode="block" size="4096" />
    <node id="CHANN25" address="0x1A000" permission="rw" mode="block" size="4096" />
    <node id="CHANN26" address="0x1B000" permission="rw" mode="block" size="4096" />
    <node id="CHANN27" address="0x1C000" permission="rw" mode="block" size="4096" />
    <node id="CHANN28" address="0x1D000" permission="rw" mode="block" size="4096" />
    <node id="CHANN29" address="0x1E000" permission="rw" mode="block" size="4096" />
    <node id="CHANN30" address="0x1F000" permission="rw" mode="block" size="4096" />
    <node id="CHANN31" address="0x20000" permission="rw" mode="block" size="4096" />
    <node id="CHANN32" address="0x21000" permission="rw" mode="block" size="4096" />
    <node id="CHANN33" address="0x22000" permission="rw" mode="block" size="4096" />
    <node id="CHANN34" address="0x23000" permission="rw" mode="block" size="4096" />
    <node id="CHANN35" address="0x24000" permission="rw" mode="block" size="4096" />
    <node id="CHANN36" address="0x25000" permission="rw" mode="block" size="4096" />
    <node id="CHANN37" address="0x26000" permission="rw" mode="block" size="4096" />
    <node id="CHANN38" address="0x27000" permission="rw" mode="block" size="4096" />
    <node id="CHANN39" address="0x28000" permission="rw" mode="block" size="4096" />
</node>
$
```

Change default IP address of the DS sender.

```
1 $ cat ipbus/Change_IP_Sen.py
2 import socket
3 import struct
4 import time
5 import numpy
6 import uhal
7
8 #Open IPbus connection using standard IP
9 hw = uhal.getDevice("sender.udp.0", "ipbusudp-2.0://192.168.102.14:50001", "file://
      sender_connection.xml")
10
11 rdIP = hw.getNode("CMD.NEWIP").read() #read current IP. An error occur if connection fails
12 hw.dispatch()
13
14 print "Sender IP Address from 192.168.102."+str(int(rdIP))
15 print "New IP..."
16 print "Last Byte = "
17 new_host    = 0
18 new_host    = int(raw_input())
19 print new_host
20
21 hw.getNode("CMD.NEWIP").write(new_host)
22 hw.dispatch()
23
24 print "Checking new IP..."
25
26 #Open IPbus connection using new IP
27 hw = uhal.getDevice("sender.udp.0", "ipbusudp-2.0://192.168.102." + str(new_host) + ":50001",
      "file://sender_connection.xml")
28
29 rdIP = hw.getNode("CMD.NEWIP").read()
30 hw.dispatch()
31
32 print "Sender IP Address changed to 192.168.102." + str(int(rdIP)) + " !"
33 $
```

Change default IP address of the DS receiver.

```
1 $ cat ipbus/Change_IP_Rec.py
2 import socket
3 import struct
4 import time
5 import numpy
6 import uhal
7
8 #Open IPbus connection using standard IP
9 hw = uhal.getDevice("receiver.udp.0", "ipbusudp-2.0://192.168.101.14:50001", "file://
      receiver_connection.xml")
10
11 rdIP = hw.getNode("CMD.NEWIP").read() #read current IP. An error occur if connection fails
12 hw.dispatch()
13
14 print "Receiver IP Address from 192.168.101."+str(int(rdIP))
15 print "New IP..."
16 print "Last Byte = "
17 new_host    = 0
18 new_host    = int(raw_input())
19 print new_host
20
21 hw.getNode("CMD.NEWIP").write(new_host)
22 hw.dispatch()
23
24 print "Checking new IP..."
25
26 #Open IPbus connection using new IP
27 hw = uhal.getDevice("receiver.udp.0", "ipbusudp-2.0://192.168.101." + str(new_host) + ":50001"
      , "file://receiver_connection.xml")
28
29 rdIP = hw.getNode("CMD.NEWIP").read()
30 hw.dispatch()
31
32 print "Receiver IP Address changed to 192.168.101." + str(int(rdIP)) + " !"
33 $
```

Write dummy words in the board sender BRAM's

```
1 $ cat ipbus/WriteDummy_Sender.py
2 import socket
3 import struct
4 import time
5 import numpy
6 import uhal
7
8 print "Sender Network is 192.168.102.0"
9 print "Type the Sender IP Last Byte: "
10 dsb_ntw = "192.168.102."
11 dsb_host = raw_input()
12
13 dsb_ip = dsb_ntw+dsb_host
14
15 #Open IPbus connection
16 hw = uhal.getDevice("sender.udp.0", "ipbusudp-2.0://localhost:10203?target="+dsb_ip+":50001",
       "file://sender_connection.xml")
17
18 print "Loading data to Sender ", dsb_ip
19
20 x=hw.getNode("CMD.RESET").read() #reset command at DSB Sender Board
21 hw.dispatch()
22 time.sleep(5) # sleep during reset
23
24 print "Please Wait .."
25 # writes dummy words on DSB Sender
26 for dsb_idx in range(40):
27     for number_samples in range(16):
28         hw.getNode("CHANN"+str(dsb_idx)).writeBlockOffset([number_samples+1], number_samples)
29         hw.dispatch()
30                                 if dsb_idx < 33:
31                                         print "Data sample", number_samples, "in GTH X0Y",(
                                            dsb_idx)
32                                 else:
33                                         print "Data sample", number_samples, "in GTH X1Y",(
                                            dsb_idx-33)
34
35 print "Please Wait ..."
36 x=hw.getNode("CMD.RESET").read() #reset command at DSB Sender Board
37 hw.dispatch()
38 time.sleep(5)
39
40 hw.getNode("CMD.COUNTER").write(0x0F) #write counter (0x0F for 16 words, and so on...)
41 hw.dispatch()
42 print "Counter set to send: 16 words"
43 print "Done! Data loaded at Sender", dsb_ip
44 $
```

Read words and check in the board receiver BRAM's

```
$ cat ipbus/ReadDummy_Sender.py
import socket
import struct
import time
import numpy
import uhal

print "Receiver Network is 192.168.101.0"
print "Type the Receiver IP Last Byte: "
dsb_ntw = "192.168.101."
dsb_host = raw_input()
dsb_ip = dsb_ntw+dsb_host

#Open IPbus connection
hw = uhal.getDevice("receiver.udp.0", "ipbusudp-2.0://localhost:10203?target="+dsb_ip+":50001"
     , "file://receiver_connection.xml")

print "Read Dummy Data from Receiver ", dsb_ip

DSFile = numpy.arange(41 * 4096)
DSFile = DSFile.reshape((41,4096))

print "Please Wait ..."
x=hw.getNode("CMD.RESET").read() #reset command at DSB Sender Board
hw.dispatch()
time.sleep(5) # sleep during reset

# writes dummy words on DSFile
dsb_idx = 1*4096
while dsb_idx <= 40*4096:
        number_samples = 0
        while number_samples <= 15:
                DSFile [dsb_idx/4096,number_samples] = number_samples+1
                number_samples = number_samples + 1
        dsb_idx = dsb_idx + 4096

print "Frame Checking"

#Reads the result at DSB Receiver
counter_errors = 0
for dsb_idx in range(40):
        print "Module: ", (dsb_idx+1)
        for number_samples in range(16):
                ml=hw.getNode("CHANN"+str(dsb_idx)).readBlockOffset(1, number_samples)
                hw.dispatch()
                if int(ml[0]) != DSFile[dsb_idx+1,number_samples]:
                        print "Sample: ", number_samples
                        print "Error:"
                        print "Expected Value:", DSFile[dsb_idx+1,number_samples]
                        print "Read Value: ", int(ml[0])
                        counter_errors = counter_errors + 1

print "32 bit words with errors at Board = ", counter_errors
$
```

Read Lantecy measured by receiver board.

```
1 $ cat ipbus/Latency_allgth.py
2 import socket
3 import struct
4 import time
5 import numpy
6
7 print "Receiver Network is 192.168.101.0"
8 print "Type the Receiver IP Last Byte: "
9 dsb_ntw = "192.168.101."
10 dsb_host = raw_input()
11 dsb_ip = dsb_ntw+dsb_host
12
13 #Open IPbus connection
14 hw = uhal.getDevice("receiver.udp.0", "ipbusudp-2.0://localhost:10203?target="+rec_ip+":50001"
       , "file://receiver_connection.xml")
15 print "Receiver Board: ", rec_ip
16
17 print "Please Wait..."
18 time.sleep(1)
19
20 for dsb_idx in range(40):
21         clk_cnt = int(receiver.getNode("CMD.Latency").readBlockOffset(1, dsb_idx))
22                 hw.dispatch()
23         latency = 4*clk_cnt # @250MHz
24         if dsb_idx < 33:
25                 print "GTH X0Y",(dsb_idx), " Latency: ", latency, " ns"
26         else:
27                 print "GTH X1Y",(dsb_idx-33), " Latency: ", latency, " ns"
28 $
```

Start the BER mode and read the BER measured by receiver board.

```
$ cat ipbus/BER_allgth.py
import socket
import struct
import time
import numpy

print "Receiver Network is 192.168.101.0"
print "Sender Network is 192.168.102.0"
print "Type the IP Last Byte: "
rec_ntw = "192.168.101."
sen_ntw = "192.168.102."
dsb_host = raw_input()

rec_ip = rec_ntw+dsb_host
sen_ip = sen_ntw+dsb_host

#Open IPbus connection
sender = uhal.getDevice("sender.udp.0", "ipbusudp-2.0://localhost:10203?target="+sen_ip+"
    :50001", "file://sender_connection.xml")
print "Sender Board: ", sen_ip

#Open IPbus connection
receiver = uhal.getDevice("receiver.udp.0", "ipbusudp-2.0://localhost:10203?target="+rec_ip+"
    :50001", "file://receiver_connection.xml")
print "Receiver Board: ", dsb_ip

# Rearm both boards
sender.getNode("CMD.REARM").read()
receiver.getNode("CMD.REARM").read()
time.sleep(1)
Start BER measurement...
sender.getNode("CMD.DSBMODE").write(0x01)
receiver.getNode("CMD.DSBMODE").write(0x01)
Process takes 7 minutes...
time.sleep(180)
3 min ...
time.sleep(180)
6 min ...
time.sleep(60)
Done!
sender.getNode("CMD.DSBMODE").write(0x00)
receiver.getNode("CMD.DSBMODE").write(0x00)
time.sleep(1)
Reading BER ...

for dsb_idx in range(40):
        ber = int(receiver.getNode("CMD.BER").readBlockOffset(1, dsb_idx))
                hw.dispatch()
        if dsb_idx < 33:
                print "GTH X0Y",(dsb_idx), " BER: E-", ber
        else:
                print "GTH X1Y",(dsb_idx-33), " BER: E-", ber
$
```

Change the values of sender tuning parameters.

```
$ cat ipbus/Tuning_Sen.py
import socket
import struct
import time
import numpy
import uhal

print "Sender Network is 192.168.102.0"
print "Type the Sender IP Last Byte: "
dsb_ntw = "192.168.102."
dsb_host = raw_input()

dsb_ip = dsb_ntw+dsb_host

#Open IPbus connection using standard IP
hw = uhal.getDevice("sender.udp.0", "chtcp-2.0://localhost:10203?target="+dsb_ip+":50001", "
    file://sender_connection.xml")

print "Tuning Board: ", dsb_ip
print "Please Wait ..."
x=hw.getNode("CMD.RESET").read() #reset command at DSB Sender Board
hw.dispatch()
time.sleep(1) # sleep during reset

dsb_idx = int(raw_input("Set DSB Module Index from 1 to 40: "),10)*16384
print hex(dsb_idx)

txpost   = int(raw_input("Set TX Post-cursor Index from 1 to 32: "),10)*512
print bin(txpost)

txpre    = int(raw_input("Set TX Pre-cursor Index from 1 to 32: "),10)*16
print bin(txpre)

txdiff   = int(raw_input("Set TX Diff Swing Index from 1 to 16: "),10)*1
print bin(txdiff)

tune_set = dsb_idx+txdiff+txpre+txpost
print "Tuning    (19 downto  0) set to: ", bin(tune_set)
print "DSB Index (19 downto 14) set to: ", bin(dsb_idx)
print "TX Post   (13 downto  9) set to: ", bin(txpost)
print "TX Post   ( 8 downto  4) set to: ", bin(txpre)
print "TX Post   ( 3 downto  0) set to: ", bin(txdiff)

hw.getNode("CMD.TUNING").write(tune_set)

time.sleep(0.5)
m = hw.getNode("CMD.TUNING").read()

print "Remote Tuning performed!"
$
```